

IoT Sederhana dengan WeMOS mini D1 untuk SMK

Onno W. Purbo
onno@indo.net.id
onno@itts.ac.id
Twitter onnowpurbo

Institut Teknologi Tangerang Selatan (ITTS)
Dinas KOMINFO Tangerang Selatan
OnnoCenter
2026

Lisensi: Karya ini dilisensikan di bawah **Creative Commons Attribution (CC BY) 4.0 International**. Anda bebas menggunakan, membagikan, dan mengadaptasi karya ini untuk tujuan apa pun, termasuk komersial, dengan tetap mencantumkan atribusi yang layak kepada penulis.

Disclaimer: Buku ini disusun untuk tujuan edukasi, dokumentasi, dan refleksi publik berdasarkan sumber-sumber yang dianggap kredibel pada saat penulisan. Penulis telah berupaya menjaga akurasi, namun sangat mungkin masih terdapat kekurangan atau kekeliruan; karena itu, buku ini tidak dimaksudkan sebagai satu-satunya rujukan yang final, dan penulis terbuka terhadap koreksi serta masukan yang membangun

IoT bukan dimulai dari cloud, tetapi dari kemampuan siswa membaca sensor, memahami rangkaian, mengirim data, dan mengendalikan perangkat secara aman melalui jaringan lokal.

Kata Pengantar

Buku “**IoT Sederhana dengan WeMOS mini D1 untuk SMK**” disusun sebagai bahan belajar praktis bagi siswa, guru, dan pemula yang ingin memahami Internet of Things dari dasar. Buku ini tidak dimulai dari teori yang rumit, tetapi dari hal yang dekat dengan praktik: menyalakan LED, membaca tombol, membaca potensiometer, menghubungkan WeMOS mini D1 ke Wi-Fi, mengirim data melalui MQTT, menjalankan broker di Ubuntu Server, dan menampilkan hasilnya di Android dashboard.

Fokus utama buku ini adalah **membangun pemahaman teknis yang sederhana, murah, terbuka, dan realistis untuk lab SMK**. Setiap bab dirancang agar siswa tidak hanya meniru rangkaian, tetapi juga memahami alur kerja IoT: dari pin, sensor, kode, jaringan, broker, topic, payload, sampai dashboard.

Semoga buku ini membantu siswa SMK membangun keberanian untuk bereksperimen, memperbaiki kesalahan, mendokumentasikan proyek, dan mengembangkan karya IoT yang lebih bermanfaat. **IoT bukan sekadar alat menyala, tetapi kemampuan membaca dunia fisik, mengirim data, mengambil keputusan, dan membangun solusi nyata.**

Jakarta, Mei 2026

Onno W. Purbo

Executive Summary

Buku “IoT Sederhana dengan WeMOS mini D1 untuk SMK” dirancang sebagai panduan praktis bagi siswa, guru, dan pemula untuk membangun sistem Internet of Things yang murah, terbuka, aman, dan realistis diterapkan di laboratorium SMK. Fokus utama buku ini adalah membawa siswa dari pemahaman paling dasar menuju proyek utuh: mengenal WeMOS mini D1, memahami pin GPIO, merangkai LED, button, dan potensiometer, membaca data analog dan digital, memakai Serial Monitor, menghubungkan board ke Wi-Fi, menjalankan Mosquitto MQTT Broker di Ubuntu Server, serta menampilkan dan mengontrol data melalui Android dashboard. Nilai strategis buku ini bukan sekadar membuat alat menyala, tetapi membentuk cara berpikir teknis: **membaca rangkaian, memahami alur data, memakai topic MQTT yang rapi, mengamankan broker dengan username/password, melakukan troubleshooting, dan mendokumentasikan hasil kerja**. Melalui proyek LED Control via Android, Button Status Monitor, Monitoring Potensiometer, Jam Digital Serial dan MQTT, hingga Smart IoT Learning Panel, siswa belajar bahwa IoT adalah integrasi antara perangkat fisik, kode, jaringan, server lokal, protokol pesan, dan antarmuka pengguna. Buku ini juga menekankan disiplin keamanan: tidak membuka broker tanpa password, tidak memberi tegangan berlebih ke pin, menjaga ground bersama, serta tidak menulis kredensial asli di laporan publik. Pada akhir pembelajaran, siswa diharapkan mampu membuat proyek akhir yang menampilkan nilai potensiometer, status tombol, kontrol LED, dan jam digital di Android melalui MQTT lokal. Lebih penting lagi, siswa mampu menjelaskan diagram, topic, payload, kode, hasil uji, masalah, solusi, dan peluang pengembangan seperti Node-RED, InfluxDB, Grafana, ESPHome, sensor suhu, sensor cahaya, relay DC, RTC, dan NTP. **Inilah fondasi IoT SMK yang tajam: sederhana untuk dimulai, cukup kuat untuk dikembangkan, dan cukup disiplin untuk dipresentasikan secara profesional.**

Daftar Isi

Kata Pengantar.....	4
Executive Summary.....	5
Daftar Isi.....	6
Kebutuhan Praktik Lab IoT WeMOS mini D1 per Siswa.....	14
1. Komponen yang Harus Disiapkan per Siswa / per Kelompok Kecil.....	14
2. Perangkat Siswa yang Perlu Disiapkan.....	14
3. Infrastruktur Lab yang Harus Disiapkan Sekolah.....	15
4. Aturan Minimum Keselamatan Praktik.....	16
BAB 1 — Mengapa Siswa SMK Perlu Belajar IoT dengan WeMOS mini D1.....	17
1.1 IoT sebagai Keterampilan Produktif Siswa SMK.....	17
1.2 Apa Itu Internet of Things Secara Sederhana.....	18
1.3 Mengapa IoT Penting untuk Siswa SMK.....	19
1.4 Contoh IoT di Sekitar Siswa.....	20
1.5 Arsitektur IoT Sederhana dengan WeMOS mini D1.....	21
1.6 Mengapa Memakai WeMOS mini D1 dan Ubuntu Server Lokal.....	23
1.7 Mengapa MQTT Cocok untuk IoT Sederhana.....	24
1.8 Peta Belajar, Etika, dan Keselamatan Berpikir IoT.....	26
BAB 2 — Mengenal WeMOS mini D1, Breadboard, Sensor, dan Dasar Listrik Aman... 28	28
2.1 Mengapa Dasar Listrik Penting Sebelum IoT.....	28
2.2 Mengenal Bentuk Fisik WeMOS mini D1.....	28
2.3 Chip ESP8266EX dan Fungsi WeMOS mini D1.....	30
2.4 Fungsi Pin Penting WeMOS mini D1.....	31
1. 3V3.....	31
2. GND.....	31
3. 5V.....	31
4. D0–D8.....	31
5. A0.....	32
6. TX/RX.....	32
7. SDA/SCL.....	32
2.5 Perbedaan WeMOS mini D1, ESP32, dan Arduino Uno.....	33
2.6 Breadboard, Jumper, Resistor, LED, Sensor, Relay, dan Servo.....	34
Breadboard.....	34
Jumper.....	35
Resistor.....	35
LED.....	35
Sensor.....	35
Relay.....	35
Servo.....	36
2.7 Tegangan 3.3V vs 5V dan Batas A0.....	37
2.8 Konsep Ground Bersama dan Cara Membaca Pinout.....	38
Ground bersama.....	38
Cara membaca pinout WeMOS mini D1.....	38

2.9 Praktik 1 — Menyalakan LED Eksternal dari WeMOS mini D1.....	40
Tujuan.....	40
Komponen.....	40
Konsep sambungan.....	40
Tabel sambungan.....	41
Kode contoh.....	41
Penjelasan kode singkat.....	41
Hasil yang diharapkan.....	42
Jika LED tidak menyala.....	42
2.10 Praktik 2 — Membaca Tombol Sederhana.....	43
Tujuan.....	43
Komponen.....	43
Konsep sederhana.....	43
Sambungan.....	43
Kode contoh.....	43
Penjelasan kode singkat.....	44
Hasil yang diharapkan.....	44
Checklist Sebelum Menyalakan Rangkaian.....	47
Latihan Bab 2.....	47
Tugas Praktik Bab 2.....	48
BAB 3 — Instalasi Arduino IDE, ESP8266 Core, dan Program Pertama.....	49
3.1 Mengapa Arduino IDE Dipakai untuk Pemula.....	49
3.2 Persiapan Ubuntu Linux dan Kabel USB Data.....	50
3.3 Instalasi Arduino IDE di Ubuntu Linux.....	51
3.4 Menambahkan ESP8266 Core melalui Boards Manager.....	52
3.5 Memilih Board LOLIN(WEMOS) D1 R2 & mini dan Port Serial.....	54
3.6 Struktur Program: setup() dan loop().....	56
setup().....	56
loop().....	57
3.7 Praktik 1 — Blink LED Internal.....	58
Tujuan.....	58
Konsep.....	58
Kode.....	58
Penjelasan Kode.....	58
3.8 Praktik 2 — Blink LED Eksternal.....	59
Tujuan.....	59
Rangkaian.....	59
Komponen.....	59
Kode.....	60
Penjelasan Kode.....	60
3.9 Praktik 3 — Menampilkan Teks di Serial Monitor.....	61
Tujuan.....	61
Kode.....	61
Penjelasan Kode.....	61

Hasil yang diharapkan.....	61
3.10 Masalah Umum dan Troubleshooting Upload.....	63
1. Port tidak muncul.....	63
2. Kabel USB hanya untuk charging.....	63
3. Driver USB serial.....	63
4. Board salah pilih.....	64
5. Upload gagal.....	64
Latihan Bab 3.....	65
Tugas Praktik Bab 3.....	65
BAB 4 — Input dan Output Dasar: Button, LED, dan Potensio.....	66
4.1 Apa Itu Input dan Output.....	66
4.2 LED sebagai Output Digital.....	67
4.3 Button sebagai Input Digital.....	69
4.4 Konsep Pull-Up dan Pull-Down.....	71
1. Pull-up.....	71
2. Pull-down.....	71
4.5 Potensiometer sebagai Input Analog.....	72
4.6 Pin A0 pada WeMOS mini D1.....	73
4.7 Membaca Nilai Analog dengan analogRead().....	74
4.8 Menampilkan Data Potensiometer ke Serial Monitor.....	75
4.9 Mengendalikan LED dengan Nilai Analog dan PWM.....	76
4.10 Kesalahan Umum Saat Membaca Tombol dan Potensio.....	78
Kesalahan pada button.....	78
Kesalahan pada potensiometer.....	78
4.11 Praktik Gabungan: LED, Button, dan Potensio.....	79
Tujuan.....	79
Fungsi sistem.....	79
Komponen.....	79
Rangkaian ringkas.....	79
Kode gabungan.....	80
Hasil yang diharapkan.....	80
4.12 Latihan, Tugas, dan Checklist Hasil.....	81
Latihan Bab 4.....	81
Tugas Praktik Bab 4.....	81
Checklist Hasil.....	81
Catatan Akhir Bab 4.....	82
BAB 5 — Jam Digital Sederhana dengan Serial Port ke Komputer.....	83
5.1 Mengapa Komunikasi Serial Penting Sebelum IoT.....	83
5.2 Apa Itu Komunikasi Serial.....	84
5.3 Fungsi TX dan RX pada WeMOS mini D1.....	85
5.4 Serial Monitor sebagai Alat Debugging.....	86
5.5 Format Data Waktu: Jam, Menit, Detik.....	88
5.6 Membuat Jam Digital Sederhana dengan millis().....	90
5.7 Mengirim Format HH:MM:SS ke Komputer.....	91

5.8 Membaca Perintah START, STOP, RESET dari Komputer.....	93
5.9 Kelemahan Jam Berbasis millis() dan Pengembangan RTC/NTP.....	96
1. RTC.....	96
2. NTP.....	96
5.10 Latihan, Tugas, dan Checklist Hasil.....	97
Latihan Bab 5.....	97
Tugas Praktik Bab 5.....	97
Checklist Hasil.....	98
Catatan Akhir Bab 5.....	98
BAB 6 — Menghubungkan WeMOS mini D1 ke Wi-Fi dan Jaringan Lokal.....	99
6.1 Mengapa Wi-Fi Penting dalam IoT.....	99
6.2 Konsep Wi-Fi pada ESP8266.....	100
6.3 Mode Station: WeMOS mini D1 Tersambung ke Router.....	101
6.4 Praktik 1 — Wi-Fi Scanner.....	102
Tujuan.....	102
Kode Wi-Fi Scanner.....	102
Hasil yang diharapkan.....	103
6.5 Praktik 2 — Koneksi ke SSID dan Password.....	104
Tujuan.....	104
Kode koneksi Wi-Fi.....	104
Penjelasan singkat.....	105
6.6 Membaca Alamat IP dan Status Koneksi.....	106
Kode status koneksi lebih jelas.....	106
6.7 Tes Koneksi Jaringan Lokal.....	108
6.8 Praktik 3 — Halaman Web Sederhana dari WeMOS mini D1.....	109
Tujuan.....	109
Kode web server sederhana.....	109
Hasil yang diharapkan.....	110
6.9 Kesalahan Umum Wi-Fi dan Troubleshooting.....	111
1. Salah password.....	111
2. SSID tidak ditemukan.....	111
3. Sinyal lemah.....	111
4. IP berubah.....	112
5. Jaringan tidak stabil.....	112
6.10 Batasan ESP8266 dan Persiapan Menuju MQTT.....	113
Latihan Bab 6.....	114
Tugas Praktik Bab 6.....	115
BAB 7 — Ubuntu Server Lokal dan Mosquitto MQTT Broker.....	116
7.1 Mengapa IoT Membutuhkan Server Lokal.....	116
7.2 Peran Ubuntu Server dalam Sistem IoT SMK.....	117
7.3 IP Statis, SSH, dan Struktur Server Lab.....	118
7.4 Instalasi Mosquitto MQTT Broker.....	119
7.5 Apa Itu MQTT.....	120
7.6 Broker, Publisher, Subscriber, Topic, Payload.....	121

1. Broker.....	121
2. Publisher.....	121
3. Subscriber.....	121
4. Topic.....	121
5. Payload.....	122
7.7 Praktik 1 — Testing mosquitto_sub dan mosquitto_pub.....	123
7.8 Username, Password, dan Larangan Broker Terbuka.....	124
7.9 Struktur Topic untuk Lab SMK.....	125
7.10 Praktik 2 — WeMOS mini D1 Publish Data Sederhana.....	127
Kode publish sederhana.....	127
Uji dari laptop.....	129
7.11 Praktik 3 — WeMOS mini D1 Subscribe Perintah ON/OFF.....	130
Kode subscribe ON/OFF.....	130
Kirim perintah dari laptop.....	132
7.12 Troubleshooting MQTT dan Checklist Hasil.....	133
Masalah 1 — Mosquitto tidak berjalan.....	133
Masalah 2 — Port 1883 tertutup.....	133
Masalah 3 — Salah username atau password.....	133
Masalah 4 — IP server salah.....	134
Masalah 5 — Topic salah.....	134
Checklist hasil Bab 7.....	134
Latihan Bab 7.....	135
Tugas Praktik Bab 7.....	135
BAB 8 — Android Dashboard untuk Monitoring dan Kontrol.....	137
8.1 Android sebagai Antarmuka IoT.....	137
8.2 Konsep Dashboard MQTT.....	138
8.3 Konfigurasi Broker MQTT di Android.....	139
8.4 Widget Monitoring Data.....	141
8.5 Tombol ON/OFF LED.....	142
8.6 Tampilan Nilai Potensiometer.....	144
8.7 Tampilan Status Button.....	146
8.8 Tampilan Jam Digital dari Topic MQTT.....	147
8.9 Payload Teks vs JSON Sederhana.....	149
8.10 Kesalahan Umum dan Checklist Hasil.....	150
Kesalahan umum konfigurasi Android dashboard.....	150
Checklist hasil Bab 8.....	150
Latihan Bab 8.....	151
Tugas Praktik Bab 8.....	151
BAB 9 — Proyek Praktik IoT Sederhana untuk SMK.....	153
9.1 Tujuan Bab 9 dan Pola Kerja Proyek.....	153
9.2 Persiapan Umum Lab IoT SMK.....	154
9.3 Standar Topic MQTT Proyek.....	155
9.4 Proyek 1 — LED Control via Android.....	156
Tujuan.....	156

Konsep.....	156
Komponen.....	156
Sambungan.....	156
Topic.....	156
Kode Inti.....	157
Output yang Diharapkan.....	157
9.5 Proyek 2 — Button Status Monitor.....	158
Tujuan.....	158
Konsep.....	158
Komponen.....	158
Sambungan dengan INPUT_PULLUP.....	158
Topic.....	158
Kode Inti.....	159
Output yang Diharapkan.....	159
9.6 Proyek 3 — Monitoring Potensiometer.....	160
Tujuan.....	160
Konsep.....	160
Komponen.....	160
Sambungan.....	160
Topic.....	161
Kode Inti.....	161
Output yang Diharapkan.....	161
9.7 Proyek 4 — Jam Digital Serial dan MQTT.....	162
Tujuan.....	162
Konsep.....	162
Komponen.....	162
Topic.....	162
Kode Inti.....	163
Output yang Diharapkan.....	163
9.8 Proyek 5 — Smart Mini Panel SMK.....	164
Tujuan.....	164
Konsep.....	164
Komponen.....	164
Sambungan.....	165
Topic Gabungan.....	165
Output yang Diharapkan.....	165
9.9 Checklist Pengujian dan Troubleshooting.....	166
Checklist Pengujian Umum.....	166
Troubleshooting Cepat.....	167
Latihan Bab 9.....	168
Tugas Praktik Bab 9.....	168
BAB 10 — Keamanan, Troubleshooting, Dokumentasi, dan Proyek Akhir.....	169
10.1 Mengapa Keamanan dan Dokumentasi Penting.....	169
10.2 Checklist Keamanan Rangkaian.....	170

10.3 Keamanan MQTT: Password, Topic, dan Akses.....	171
10.4 Backup Konfigurasi Mosquitto dan Kode Proyek.....	172
10.5 Troubleshooting WeMOS mini D1.....	173
1. Gagal upload.....	173
2. Port tidak muncul.....	173
3. Wi-Fi gagal konek.....	173
4. MQTT tidak tersambung.....	174
5. Sensor tidak terbaca.....	174
10.6 Troubleshooting Ubuntu Server.....	175
1. IP berubah.....	175
2. Mosquitto mati.....	175
3. Firewall menutup port.....	175
4. Password salah.....	176
5. Broker tidak bisa diakses dari perangkat lain.....	176
10.7 Troubleshooting Android Dashboard.....	177
10.8 Format Laporan dan Presentasi Proyek SMK.....	179
10.9 Proyek Akhir — Smart IoT Learning Panel dengan WeMOS mini D1.....	180
10.10 Ide Pengembangan Lanjutan.....	182
Latihan Bab 10.....	183
Tugas Praktik Bab 10.....	183
Lampiran A — Daftar Komponen Praktik.....	185
Lampiran B — Gambar Sambungan WeMOS mini D1.....	186
B.1 WeMOS mini D1 ke LED.....	186
B.2 WeMOS mini D1 ke Button.....	186
B.3 WeMOS mini D1 ke Potensiometer.....	187
B.4 WeMOS mini D1 ke Relay DC.....	187
B.5 WeMOS mini D1 ke Serial Komputer.....	187
Lampiran C — Pinout WeMOS mini D1.....	188
Pin yang Perlu Hati-Hati Saat Boot.....	189
Lampiran D — Perintah Ubuntu Server dan Mosquitto.....	190
D.1 Update Sistem.....	190
D.2 Instalasi Mosquitto.....	190
D.3 Cek Service Mosquitto.....	190
D.4 Subscribe Topic.....	191
D.5 Publish Pesan.....	191
D.6 MQTT dengan Username dan Password.....	191
D.7 Firewall Sederhana.....	192
Lampiran E — Kode Lengkap.....	193
E.1 Blink LED Internal.....	193
E.2 Blink LED Eksternal.....	194
E.3 Button.....	195
E.4 Potensiometer.....	196
E.5 Serial Clock.....	196
E.6 Wi-Fi Connect.....	198

E.7 MQTT Publish.....	199
E.8 MQTT Subscribe.....	201
E.9 Proyek Akhir — Smart IoT Learning Panel.....	204
Lampiran F — Checklist Troubleshooting.....	209
F.1 Upload Program.....	209
F.2 Serial Port.....	209
F.3 Wi-Fi.....	209
F.4 MQTT.....	209
F.5 Android Dashboard.....	210
F.6 Sensor.....	210
F.7 Power.....	210
F.8 Ground.....	210
Lampiran G — Template Laporan Proyek IoT SMK.....	211
Judul Proyek.....	211
1. Tujuan.....	211
2. Komponen.....	211
3. Diagram.....	212
4. Topic MQTT.....	212
5. Kode.....	212
6. Hasil Uji.....	212
7. Masalah.....	213
8. Solusi.....	213
9. Kesimpulan.....	213
Glossary IoT Dasar.....	214
Referensi.....	218
Tentang Penulis.....	220

Kebutuhan Praktik Lab IoT WeMOS mini D1 per Siswa

1. Komponen yang Harus Disiapkan per Siswa / per Kelompok Kecil

Komponen	Jumlah Minimal	Fungsi
WeMOS mini D1 / LOLIN D1 mini	1 unit	Board utama berbasis ESP8266 untuk membaca input, mengendalikan output, dan terhubung ke Wi-Fi.
Breadboard	1 unit	Media merangkai komponen tanpa solder.
Kabel jumper male-male	10–20 buah	Menghubungkan WeMOS, breadboard, sensor, dan aktuator.
LED	2–3 buah	Output digital sederhana untuk indikator ON/OFF.
Resistor 220Ω / 330Ω	3–5 buah	Pembatas arus LED agar pin WeMOS aman.
Push button	1–2 buah	Input digital untuk membaca status ditekan/dilepas.
Potensiometer	1 buah	Input analog untuk pin A0.
Relay module DC	1 buah, opsional	Latihan kontrol beban DC kecil. Tidak untuk listrik 220V tanpa pengawasan guru.
Buzzer	1 buah, opsional	Output suara sederhana untuk alarm/indikator.
Kabel USB data	1 buah	Upload program dan komunikasi serial. Wajib kabel data, bukan kabel charger saja.

Komponen inti yang disebutkan dalam Lampiran A buku adalah WeMOS mini D1, breadboard, kabel jumper, LED, resistor, push button, potensiometer, relay module DC, buzzer, dan kabel USB data. Catatan pentingnya: **gunakan kabel USB data**, karena kabel charger sering membuat board menyala tetapi port serial tidak muncul.

2. Perangkat Siswa yang Perlu Disiapkan

Setiap siswa atau kelompok idealnya memiliki:

Laptop / PC Ubuntu Linux untuk menjalankan Arduino IDE, upload program, Serial Monitor, dan uji MQTT sederhana.

HP Android untuk membuat dashboard monitoring dan kontrol melalui MQTT.

Akses Wi-Fi lab agar WeMOS mini D1, laptop, Android, dan server MQTT berada dalam jaringan lokal yang sama.

3. Infrastruktur Lab yang Harus Disiapkan Sekolah

Infrastruktur Lab	Kebutuhan Minimal
Jaringan Wi-Fi lokal	Satu SSID khusus lab, stabil, dapat diakses laptop, Android, dan WeMOS mini D1.
Router / access point	Mendukung koneksi banyak perangkat siswa secara bersamaan.
Ubuntu Server lokal	PC bekas, mini PC, atau server lab untuk menjalankan Mosquitto MQTT Broker.
Mosquitto MQTT Broker	Sudah aktif, dapat diakses dari jaringan lokal, memakai username dan password.
Alamat IP broker tetap	IP server harus diketahui siswa dan sebaiknya dibuat statis.
Akun MQTT praktik	Username/password untuk siswa atau kelompok.
Port MQTT 1883 terbuka di jaringan lokal	Firewall tidak boleh memblokir akses dari laptop, Android, dan WeMOS.
Arduino IDE + ESP8266 Core	Terpasang di laptop lab atau laptop siswa.
Aplikasi Android dashboard MQTT	Dipasang di HP siswa untuk subscribe/publish topic MQTT.
Stopkontak dan kabel listrik aman	Cukup untuk laptop dan server, tidak membebani satu titik listrik.
Meja praktek elektronik	Cukup luas untuk laptop, breadboard, komponen, dan catatan.
Prosedur keselamatan lab	Aturan tegangan 3.3V/5V, GND bersama, larangan beban 220V, dan pengecekan sebelum menyalakan rangkaian.

4. Aturan Minimum Keselamatan Praktik

Jangan langsung memberi tegangan 5V ke pin GPIO WeMOS mini D1. Board bekerja pada logika 3.3V dan A0 memiliki batas tegangan.

LED wajib memakai resistor. LED tanpa resistor dapat merusak pin board.

Semua perangkat yang saling bertukar sinyal harus punya GND bersama.

Jangan memakai beban listrik 220V dalam praktik dasar. Untuk siswa SMK, praktek cukup memakai LED, button, potensiometer, buzzer, dan relay DC kecil di bawah pengawasan guru.

Broker MQTT tidak boleh terbuka tanpa password. Buku menekankan penggunaan username/password dan struktur topic yang rapi agar praktik aman dan mudah diuji.

BAB 1 — Mengapa Siswa SMK Perlu Belajar IoT dengan WeMOS mini D1

1.1 IoT sebagai Keterampilan Produktif Siswa SMK

Siswa SMK tidak cukup hanya bisa memakai perangkat. Siswa SMK perlu mampu **membaca kondisi nyata, mengirim data, memahami jaringan, dan mengendalikan perangkat.**

Di masa sekarang, banyak sistem sederhana sudah bergerak ke arah otomatisasi. Ruang kelas dapat dipantau suhunya. Lampu dapat dikontrol dari HP. Tombol dapat mengirim status ke server. Potensiometer dapat mengirim nilai analog. Jam digital dapat mengirim data melalui *serial port* atau MQTT. Semua ini adalah pintu masuk ke IoT.

IoT membuat siswa SMK belajar lintas bidang:

- **Elektronika dasar.**
- **Pemrograman mikrokontroler.**
- **Jaringan Wi-Fi.**
- **Server lokal.**
- **Protokol komunikasi.**
- **Dashboard Android.**
- **Keamanan sistem sederhana.**
- **Dokumentasi proyek teknis.**

Bagi siswa SMK, IoT bukan sekadar tren. IoT adalah **keterampilan kerja**. Siswa yang bisa membuat alat membaca sensor, mengirim data ke MQTT Broker, dan menampilkannya di Android sudah memiliki dasar kuat untuk dunia otomasi, jaringan, elektronika, dan industri kecil.

Kalimat kunci:

IoT adalah cara siswa SMK menghubungkan dunia fisik dengan dunia digital.

1.2 Apa Itu Internet of Things Secara Sederhana

Internet of Things adalah konsep ketika benda fisik diberi kemampuan untuk **merasakan kondisi**, **mengirim data**, dan kadang **menerima perintah** melalui jaringan.

Benda fisik itu bisa berupa:

- LED.
- Tombol.
- Potensiometer.
- Sensor suhu.
- Relay.
- Buzzer.
- Jam digital.
- Panel kontrol sederhana.
- Mesin kecil.
- Alat praktik di sekolah.

Dalam IoT, benda tidak lagi berdiri sendiri. Benda menjadi bagian dari sistem. Data dari benda dapat dikirim ke server. Server dapat meneruskan data ke dashboard. Pengguna dapat melihat kondisi perangkat melalui HP Android.

Secara sederhana, IoT bekerja dengan pola:

Sense → Send → Store / Route → Show → Act

Artinya:

- **Sense:** sensor membaca kondisi.
- **Send:** board mengirim data.
- **Store / Route:** server menerima atau meneruskan data.
- **Show:** dashboard menampilkan data.
- **Act:** aktuator menjalankan aksi.

Contoh sederhana:

```
Button ditekan
    ↓
WeMOS mini D1 membaca status
    ↓
Data dikirim lewat Wi-Fi
    ↓
MQTT Broker menerima pesan
    ↓
Android menampilkan status tombol
```

Inti IoT:

Data kecil dari alat sederhana dapat menjadi informasi yang berguna jika dikirim, dibaca, dan ditampilkan dengan benar.

1.3 Mengapa IoT Penting untuk Siswa SMK

IoT penting karena menggabungkan banyak kompetensi yang dekat dengan dunia SMK. Siswa tidak hanya belajar teori jaringan atau elektronika secara terpisah. Siswa belajar membuat **sistem utuh**.

Dalam satu proyek IoT sederhana, siswa belajar:

- Menentukan masalah.
- Memilih sensor.
- Memahami tegangan.
- Membaca pin board.
- Menulis program.
- Menghubungkan board ke Wi-Fi.
- Mengirim data ke server.
- Membaca data di Android.
- Menguji sistem.
- Mencatat error.
- Membuat laporan.

Ini membuat IoT sangat cocok untuk pembelajaran berbasis proyek. Siswa dapat melihat hasil langsung. LED menyala. Nilai potensiometer berubah. Status tombol muncul di dashboard. Jam digital tampil di komputer. Data mengalir melalui jaringan.

Namun IoT juga perlu dipahami secara kritis. Tidak semua alat harus dibuat IoT. Jika cukup memakai saklar biasa, jangan dipaksa menjadi sistem rumit. IoT berguna ketika data perlu:

- Dipantau.
- Dikirim.
- Dicatat.
- Dikendalikan.
- Ditampilkan.
- Dianalisis.
- Diakses dari perangkat lain.

Prinsip kritis:

IoT yang baik menyelesaikan masalah nyata, bukan sekadar membuat alat terlihat modern.

1.4 Contoh IoT di Sekitar Siswa

IoT tidak harus dimulai dari proyek mahal. IoT bisa dimulai dari masalah kecil yang dekat dengan siswa.

Di rumah, IoT dapat dipakai untuk:

- Monitoring suhu kamar.
- Kontrol lampu kecil.
- Alarm pintu.
- Monitoring tangki air.
- Notifikasi tombol darurat sederhana.

Di sekolah, IoT dapat dipakai untuk:

- Monitoring suhu ruang kelas.
- Status pintu laboratorium.
- Panel kontrol LED praktik.
- Dashboard nilai potensiometer.
- Jam digital kelas yang dikirim ke dashboard.
- Monitoring ruang server kecil.

Di bengkel, IoT dapat dipakai untuk:

- Peringatan suhu mesin.
- Status alat menyala atau mati.
- Tombol laporan kondisi alat.
- Penghitung proses sederhana.

Di pertanian, IoT dapat dipakai untuk:

- Monitoring kelembaban tanah.
- Kontrol pompa kecil.
- Monitoring suhu greenhouse.
- Peringatan tangki air kosong.

Di keamanan, IoT dapat dipakai untuk:

- Tombol darurat.
- Alarm pintu.
- Sensor gerak.
- Status lampu keamanan.

Di industri kecil, IoT dapat dipakai untuk:

- Monitoring suhu gudang.
- Monitoring freezer.
- Status mesin produksi.
- Peringatan kondisi lingkungan kerja.

Semua contoh di atas dapat dimulai dari sistem kecil. Tidak perlu langsung membuat sistem besar. Yang penting siswa memahami **alur data** dan **fungsi setiap komponen**.

Kalimat penting:

Proyek IoT terbaik untuk siswa SMK adalah proyek kecil yang jelas masalahnya, murah komponennya, dan bisa dijelaskan alur kerjanya.

1.5 Arsitektur IoT Sederhana dengan WeMOS mini D1

Arsitektur utama buku ini adalah:

```
Sensor → WeMOS mini D1 → Wi-Fi → MQTT Broker → Android /  
Dashboard
```

Setiap bagian memiliki peran.

Sensor membaca kondisi dunia nyata. Dalam buku ini, sensor dan input sederhana dapat berupa button, potensiometer, atau sensor lain yang mudah dipahami.

WeMOS mini D1 membaca input, memproses data sederhana, lalu mengirim data melalui Wi-Fi. WeMOS mini D1 cocok untuk pembelajaran karena kecil, murah, dan berbasis ESP8266EX. Dokumentasi LOLIN menyebut D1 mini memiliki tegangan operasi 3.3V, 11 pin digital I/O, 1 input analog dengan batas 3.2V, clock 80/160 MHz, dan flash 4 MB. ([Wemos](#))

Wi-Fi menjadi jalur komunikasi. Tanpa Wi-Fi, data hanya berhenti di board. Dengan Wi-Fi, data dapat dikirim ke server lokal.

MQTT Broker menjadi pusat pertukaran pesan. Perangkat tidak perlu saling berbicara langsung. WeMOS mini D1 cukup mengirim pesan ke topic tertentu. Android cukup membaca topic yang dibutuhkan.

Android / Dashboard menjadi tampilan untuk manusia. Data yang tadinya hanya angka dapat dilihat sebagai status, nilai, tombol, atau indikator.

Contoh alur monitoring:

```
Potensiometer diputar
↓
WeMOS mini D1 membaca nilai A0
↓
WeMOS mini D1 publish ke topic smk/lab1/potensio
↓
Mosquitto menerima pesan
↓
Android subscribe topic tersebut
↓
Nilai potensiometer tampil di HP
```

Contoh alur kontrol:

```
Android mengirim ON
↓
Mosquitto menerima pesan di topic smk/lab1/led/set
↓
WeMOS mini D1 subscribe topic tersebut
↓
LED menyala
```

Pola berpikir arsitektur:

Sensor membaca, board mengirim, broker mengatur, dashboard menampilkan, aktuator menjalankan aksi.

1.6 Mengapa Memakai WeMOS mini D1 dan Ubuntu Server Lokal

WeMOS mini D1 dipilih karena sederhana dan cocok untuk pemula. Board ini kecil, murah, dan langsung memiliki Wi-Fi. Untuk siswa SMK, ini penting karena pembelajaran IoT tidak perlu dimulai dari perangkat mahal.

Keunggulan WeMOS mini D1:

- Kecil dan murah.
- Sudah memiliki Wi-Fi.
- Cocok untuk sensor sederhana.
- Mendukung Arduino Core untuk ESP8266.
- Bisa dipakai untuk MQTT.
- Cocok untuk proyek kelas dan lab sekolah.
- Cukup untuk LED, button, potensiometer, relay DC kecil, dan dashboard sederhana.

ESP8266 Arduino Core memungkinkan ESP8266 diprogram memakai pola program Arduino dan menyediakan pustaka untuk Wi-Fi, digital I/O, analog input, serial, timing, SPI, I2C, dan fitur lain. ([Esp8266 Arduino Core Documentation](#))

Namun WeMOS mini D1 punya batas. Pin lebih sedikit dibanding board yang lebih besar. Input analog hanya satu. Logika bekerja pada 3.3V. Karena itu, siswa harus disiplin membaca pinout, memahami tegangan, dan tidak asal menyambungkan sensor 5V.

Ubuntu Server lokal dipakai agar siswa belajar IoT sebagai sistem lengkap. Server lokal dapat menjalankan Mosquitto MQTT Broker, menerima pesan, mengatur komunikasi, dan menjadi pusat pembelajaran jaringan.

Keuntungan Ubuntu Server lokal:

- Tidak tergantung layanan luar.
- Cocok untuk jaringan sekolah.
- Dapat memakai PC bekas atau mini PC.
- Mengajarkan IP address, SSH, port, service, dan firewall.
- Mudah dikembangkan ke Node-RED, InfluxDB, atau Grafana.
- Mendukung pembelajaran open source secara nyata.

Prinsip desain:

WeMOS mini D1 membaca dan mengirim data. Ubuntu Server mengatur layanan. Android menampilkan hasil.

1.7 Mengapa MQTT Cocok untuk IoT Sederhana

MQTT cocok untuk IoT karena ringan, sederhana, dan memakai pola *publish/subscribe*. MQTT adalah standar OASIS untuk *messaging protocol* IoT yang sangat ringan dan dirancang untuk perangkat dengan sumber daya kecil serta jaringan dengan bandwidth terbatas. ([MQTT](#))

Konsep MQTT mudah dipahami:

- **Broker:** pusat pertukaran pesan.
- **Publisher:** perangkat yang mengirim pesan.
- **Subscriber:** perangkat yang menerima pesan.
- **Topic:** alamat pesan.
- **Payload:** isi pesan.

Contoh:

```
Topic    : smk/lab1/button
Payload  : PRESSED
```

Artinya, WeMOS mini D1 mengirim status tombol ke topic `smk/lab1/button`.

Contoh lain:

```
Topic    : smk/lab1/led/set
Payload  : ON
```

Artinya, Android mengirim perintah agar LED menyala.

Dalam buku ini, broker MQTT yang digunakan adalah **Eclipse Mosquitto**. Mosquitto adalah broker MQTT open source yang mendukung MQTT 5.0, 3.1.1, dan 3.1, serta cocok untuk perangkat kecil sampai server penuh. ([Eclipse Mosquitto](#))

MQTT cocok untuk siswa SMK karena:

- Konsepnya sederhana.
- Ringan untuk WeMOS mini D1.
- Cocok untuk data kecil.
- Mudah diuji.
- Cocok untuk Android dashboard.
- Cocok untuk jaringan lokal sekolah.
- Banyak dipakai dalam sistem IoT nyata.

Namun MQTT harus dipakai dengan aman. Broker tanpa password dapat dibaca atau dikirim pesan oleh pihak yang tidak berhak. Karena itu, siswa harus sejak awal mengenal **username**, **password**, dan **struktur topic yang rapi**.

Contoh topic yang rapi:

```
smk/lab1/led/set  
smk/lab1/led/status  
smk/lab1/button  
smk/lab1/potensio  
smk/lab1/jam
```

Prinsip keamanan awal:

Jangan biarkan broker MQTT terbuka tanpa kontrol.

1.8 Peta Belajar, Etika, dan Keselamatan Berpikir IoT

Buku ini disusun bertahap. Siswa tidak langsung masuk ke proyek akhir. Siswa belajar dari konsep, lalu perangkat, lalu kode, lalu jaringan, lalu server, lalu dashboard.

Peta belajar buku:

- 1. Memahami konsep IoT**
 - Apa itu IoT.
 - Mengapa IoT penting.
 - Bagaimana data bergerak dari sensor ke dashboard.
- 2. Mengenal WeMOS mini D1 dan dasar listrik**
 - Pin.
 - Tegangan.
 - Ground.
 - Breadboard.
 - Sensor sederhana.
- 3. Memprogram WeMOS mini D1**
 - Arduino IDE.
 - ESP8266 Core.
 - `setup()`.
 - `loop()`.
 - Blink.
 - Serial Monitor.
- 4. Membaca input dan output dasar**
 - LED.
 - Button.
 - Potensiometer.
 - A0.
 - PWM.
- 5. Membuat jam digital serial**
 - Serial port.
 - Format waktu.
 - Komunikasi ke komputer.
 - Perintah `START`, `STOP`, `RESET`.
- 6. Menghubungkan ke Wi-Fi**
 - SSID.
 - Password.
 - IP address.
 - Koneksi lokal.
- 7. Membangun MQTT Broker**
 - Ubuntu Server.
 - Mosquitto.
 - Publish.
 - Subscribe.
 - Topic.
 - Payload.

8. Membuat Android dashboard

- Monitoring data.
- Tombol kontrol.
- Status perangkat.
- Tampilan jam digital.

9. Mengerjakan proyek praktik

- LED control.
- Button monitor.
- Potensiometer monitor.
- Jam digital serial dan MQTT.
- Smart Mini Panel SMK.

10. Menyelesaikan proyek akhir

- Desain.
- Diagram.
- Kode.
- Pengujian.
- Troubleshooting.
- Presentasi.

Etika IoT perlu dikenalkan sejak awal. IoT dapat membaca data lingkungan dan kondisi pengguna. Karena itu, siswa harus belajar membuat sistem yang **aman, jujur**, dan **bertanggung jawab**.

Etika dasar IoT:

- Jangan mengambil data tanpa izin.
- Jangan memasang sensor untuk memata-matai orang.
- Jangan menyebarkan alamat IP, username, atau password.
- Jangan membuka broker MQTT tanpa password.
- Jangan mengontrol perangkat berbahaya tanpa pengawasan.
- Jangan membuat sistem yang tidak bisa dijelaskan cara kerjanya.

Keselamatan juga bagian dari cara berpikir IoT. Kesalahan kecil seperti salah tegangan, salah pin, salah *ground*, atau salah beban relay dapat merusak alat dan membahayakan pengguna.

Penutup Bab 1:

Siswa SMK yang belajar IoT harus berpikir seperti teknisi muda: paham alat, paham data, paham jaringan, paham risiko, dan mampu menjelaskan keputusan teknisnya.

BAB 2 — Mengenal WeMOS mini D1, Breadboard, Sensor, dan Dasar Listrik Aman

2.1 Mengapa Dasar Listrik Penting Sebelum IoT

IoT tidak dimulai dari internet. IoT dimulai dari **rangkaian yang benar**.

Banyak kegagalan pemula bukan karena kode salah, tetapi karena kabel salah, pin salah, tegangan salah, atau *ground* tidak tersambung. WeMOS mini D1 adalah board kecil. Ia kuat untuk belajar IoT, tetapi tetap sensitif terhadap salah sambung.

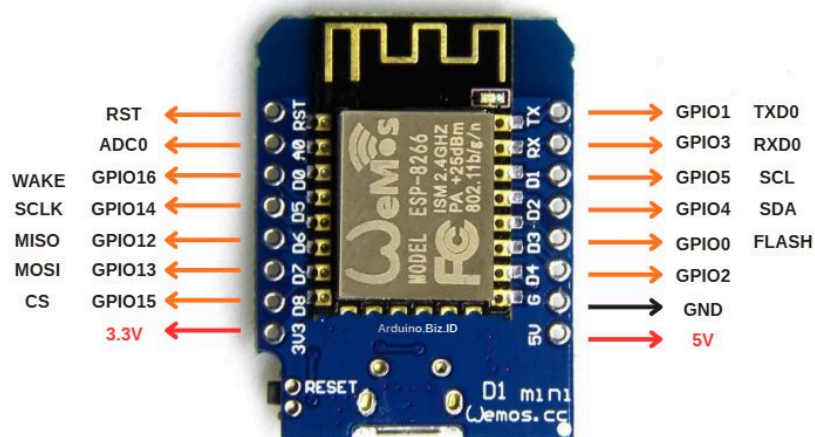
Sebelum membuat dashboard Android, MQTT, dan server lokal, siswa harus memahami:

- Pin mana untuk daya.
- Pin mana untuk ground.
- Pin mana untuk sinyal.
- Sensor bekerja di 3.3V atau 5V.
- A0 hanya untuk input analog.
- GPIO tidak boleh menerima sinyal sembarangan.
- Relay dan servo tidak boleh mengambil arus besar langsung dari board.

Kalimat penting:

IoT yang gagal sering dimulai dari kabel yang dianggap sepele.

2.2 Mengenal Bentuk Fisik WeMOS mini D1



Gambar 2.1 Pinout WeMOS mini D1 sebagai peta utama sebelum menyambungkan sensor dan aktuator.

WeMOS mini D1, atau LOLIN D1 mini, adalah board kecil berbasis **ESP8266EX**. Ukurannya kecil, mudah ditancapkan ke breadboard, dan cocok untuk praktik IoT dasar di SMK.

Dokumentasi resmi LOLIN menyebut D1 mini memiliki **operating voltage 3.3V**, **11 digital I/O**, **1 analog input dengan batas 3.2V maksimum**, clock **80/160 MHz**, flash **4 MB**, ukuran **34.2 × 25.6 mm**, dan berat sekitar **3 gram**. ([Wemos](#))

Bagian fisik penting:

- **Port USB**
Untuk upload program dan memberi daya saat praktik.
- **Chip USB-to-serial**
Menghubungkan komputer Ubuntu Linux ke WeMOS mini D1 melalui *serial port*.
- **Modul ESP8266EX**
Otak utama board, menangani program, Wi-Fi, dan I/O.
- **Pin 3V3**
Jalur tegangan 3.3V untuk modul dan sensor tertentu.
- **Pin 5V**
Jalur daya dari USB atau sumber 5V pada board.
- **Pin GND**
Referensi nol volt untuk seluruh rangkaian.
- **Pin D0–D8**
Pin digital untuk input-output.
- **Pin A0**
Input analog.
- **Pin TX/RX**
Komunikasi serial.

Catatan kritis:

Nama pin D0–D8 bukan selalu sama dengan nomor GPIO ESP8266. Selalu cek pinout board yang digunakan.

2.3 Chip ESP8266EX dan Fungsi WeMOS mini D1

WeMOS mini D1 memakai chip **ESP8266EX**. Chip ini membuat board kecil dapat terhubung ke Wi-Fi tanpa modul tambahan.

Dokumentasi Espressif menjelaskan ESP8266EX sebagai solusi **Wi-Fi SoC** yang sangat terintegrasi untuk kebutuhan daya efisien, desain ringkas, dan kinerja yang andal. ([Espressif Documentation](#))

Fungsi WeMOS mini D1 dalam buku ini:

- Membaca **button**.
- Menyalakan **LED**.
- Membaca **potensiometer** melalui A0.
- Mengirim data ke komputer melalui *serial port*.
- Menghubungkan diri ke Wi-Fi.
- Mengirim data ke MQTT Broker.
- Menerima perintah dari Android.
- Menjadi node kecil dalam sistem IoT lokal.

Namun WeMOS mini D1 bukan komputer besar. Ia cocok sebagai **node sensor dan aktuator**, bukan sebagai server utama. Server tetap diletakkan di Ubuntu Server lokal.

Prinsip desain:

WeMOS mini D1 membaca dan mengirim data. Ubuntu Server mengatur layanan. Android menampilkan hasil.

2.4 Fungsi Pin Penting WeMOS mini D1

Pemula tidak perlu menghafal semua detail internal ESP8266. Yang penting adalah memahami fungsi pin yang sering dipakai.

1. 3V3

Pin **3V3** menyediakan tegangan 3.3V.

Gunakan untuk sensor atau modul yang bekerja pada 3.3V. WeMOS mini D1 sendiri bekerja pada logika 3.3V.

Aturan aman:

Jangan memasukkan sinyal 5V langsung ke pin I/O WeMOS mini D1.

2. GND

GND adalah *ground*. Semua komponen yang saling bertukar sinyal harus memiliki GND bersama.

Tanpa GND bersama, data bisa kacau. LED bisa tidak menyala. Tombol bisa terbaca salah. Sensor bisa memberi nilai tidak stabil.

3. 5V

Pin **5V** biasanya berasal dari USB atau jalur input daya board.

Pin ini bukan berarti semua pin WeMOS aman menerima sinyal 5V. Daya 5V boleh ada pada jalur daya, tetapi sinyal ke pin digital tetap harus diperhatikan.

Kalimat penting:

5V untuk daya bukan berarti GPIO aman menerima 5V.

4. D0–D8

Pin **D0–D8** adalah pin digital pada board.

Digunakan untuk:

- LED.
- Button.
- Buzzer.
- Relay module.
- Sinyal sensor digital.
- Kontrol sederhana.

Namun beberapa pin punya fungsi khusus saat *boot*. Karena itu, untuk pemula, pilih pin yang umum aman untuk praktik sederhana seperti **D1, D2, D5, D6, D7**.

5. A0

Pin **A0** adalah input analog. Ini penting untuk membaca potensiometer.

LOLIN D1 mini memiliki **1 analog input** dengan batas **3.2V maksimum** menurut dokumentasi resminya. ([Wemos](#))

Aturan keras:

Jangan memasukkan tegangan di atas batas A0.

6. TX/RX

Pin **TX** dan **RX** dipakai untuk komunikasi serial.

- **TX** berarti *transmit*, mengirim data.
- **RX** berarti *receive*, menerima data.

Dalam buku ini, TX/RX penting untuk materi **jam digital sederhana dengan serial port ke komputer** pada Bab 5. Outline buku memang menempatkan komunikasi serial sebagai materi khusus sebelum MQTT.

7. SDA/SCL

SDA dan **SCL** dipakai untuk komunikasi I2C.

- **SDA** membawa data.
- **SCL** membawa clock.

Pada praktik sederhana, I2C belum wajib. Tetapi siswa perlu mengenalnya karena banyak sensor digital, OLED, dan RTC memakai I2C.

2.5 Perbedaan WeMOS mini D1, ESP32, dan Arduino Uno

WeMOS mini D1, ESP32, dan Arduino Uno sama-sama bisa dipakai belajar mikrokontroler. Namun perannya berbeda.

Arduino Uno Rev3 adalah board berbasis ATmega328P dengan **14 digital input/output pins**, **6 analog inputs**, dan clock **16 MHz**. ([Dokumentasi Arduino](#))

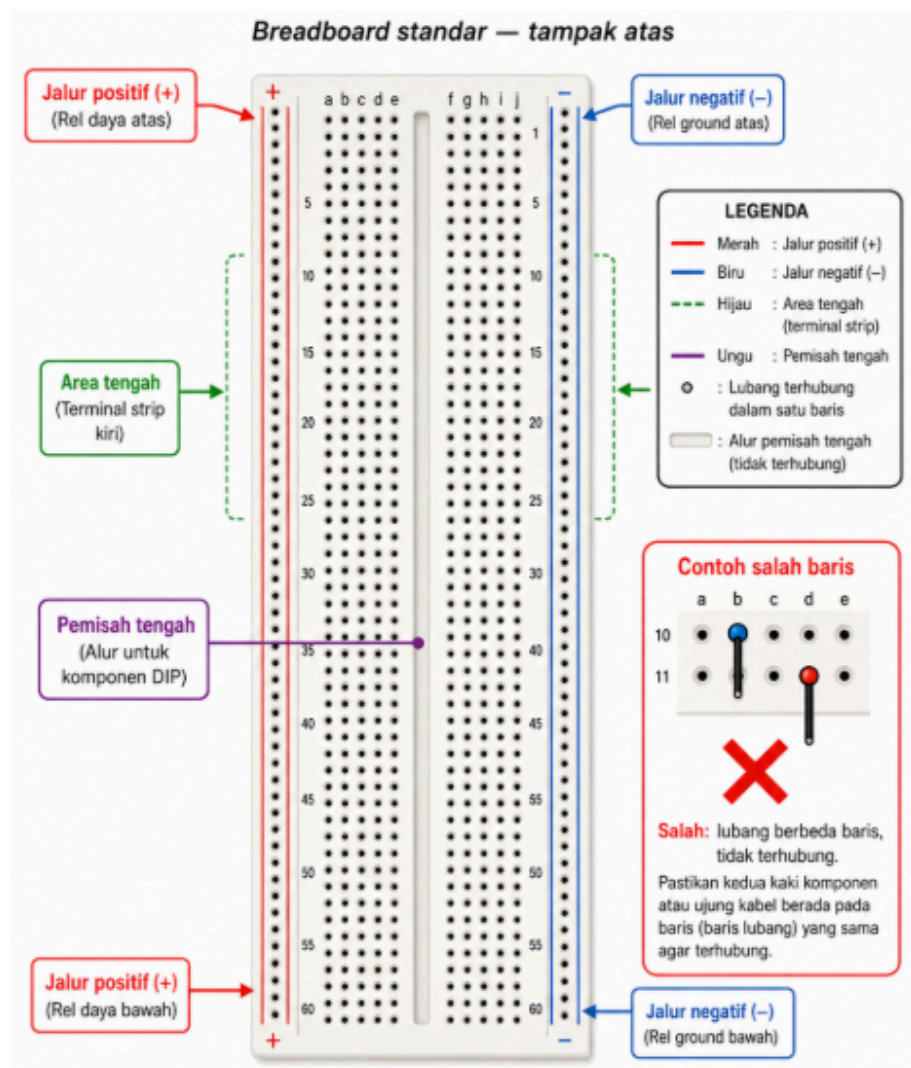
Perbandingan sederhana:

Aspek	WeMOS mini D1	ESP32	Arduino Uno
Chip utama	ESP8266EX	ESP32	ATmega328P
Wi-Fi	Ada	Ada	Tidak bawaan
Bluetooth	Tidak	Ada pada banyak varian	Tidak bawaan
Logika umum	3.3V	3.3V	5V
Digital I/O	11	Lebih banyak	14
Analog input	1	Lebih banyak	6
Cocok untuk IoT murah	Sangat cocok	Sangat cocok	Perlu modul tambahan
Cocok untuk pemula	Cocok, asal paham pin	Cocok, lebih kompleks	Cocok untuk dasar elektronik
Risiko utama	Pin terbatas, A0 terbatas	Salah pilih pin, 3.3V	Tidak punya Wi-Fi bawaan

Pesan strategis:

WeMOS mini D1 cocok untuk IoT sederhana karena kecil, murah, dan sudah memiliki Wi-Fi.

2.6 Breadboard, Jumper, Resistor, LED, Sensor, Relay, dan Servo



Gambar 2.2 Breadboard memudahkan percobaan tanpa solder, tetapi jalurnya harus dipahami agar tidak salah sambung.

Breadboard

Breadboard adalah papan percobaan tanpa solder. Siswa bisa memasang dan membongkar komponen dengan cepat.

Kesalahan umum breadboard:

- Kabel masuk baris yang salah.
- Jalur positif dan negatif tertukar.
- Komponen tidak menancap penuh.
- GND tidak tersambung.
- Kaki LED terbalik.

Jumper

Kabel jumper menghubungkan WeMOS mini D1 ke breadboard.

Gunakan warna konsisten:

- **Merah** untuk VCC.
- **Hitam / biru** untuk GND.
- **Kuning / hijau** untuk sinyal.

Resistor

Resistor membatasi arus. Pada LED, resistor wajib dipakai.

Untuk praktik LED, gunakan:

- **220 Ω** , atau
- **330 Ω** .

LED

LED adalah output paling sederhana.

- Kaki panjang: anoda / positif.
- Kaki pendek: katoda / negatif.

Aturan:

LED jangan langsung disambungkan tanpa resistor.

Sensor

Sensor membaca kondisi nyata. Pada bab awal, sensor sederhana dapat berupa:

- Button sebagai input digital.
- Potensiometer sebagai input analog.
- LDR untuk cahaya pada pengembangan berikutnya.
- Sensor suhu untuk proyek lanjutan.

Relay

Relay adalah saklar elektronik untuk mengendalikan beban.

Untuk pemula, gunakan beban DC rendah. Jangan langsung memakai listrik berbahaya.

Aturan keras:

Jangan praktik beban 220V tanpa guru, prosedur keselamatan, dan modul yang benar.

Servo

Servo adalah aktuator yang bergerak ke sudut tertentu. Servo biasanya butuh arus lebih besar daripada pin board.

Aturan aman:

Jangan memberi daya servo besar langsung dari pin WeMOS mini D1.

2.7 Tegangan 3.3V vs 5V dan Batas A0

WeMOS mini D1 bekerja pada logika **3.3V**. Ini bagian paling penting dalam bab ini.

Masalah umum pemula:

- Mengira semua board sama.
- Mengira semua sensor 5V aman.
- Menghubungkan sinyal 5V ke D0–D8.
- Memasukkan tegangan terlalu tinggi ke A0.
- Mengambil arus besar dari pin board.
- Menghubungkan relay atau servo tanpa catu daya yang sesuai.

Aturan aman tegangan:

- Gunakan sensor 3.3V jika memungkinkan.
- Cek datasheet sensor.
- Jangan masukkan sinyal 5V ke pin digital.
- Perhatikan batas A0.
- Gunakan pembagi tegangan atau *level shifter* jika diperlukan.
- Satukan GND jika memakai catu daya eksternal.

A0 dan potensiometer

A0 dipakai untuk membaca tegangan analog. Dalam buku ini, A0 digunakan untuk membaca potensiometer.

Potensiometer dapat dipahami sebagai pembagi tegangan. Saat diputar, nilai tegangan di kaki tengah berubah. WeMOS mini D1 membaca perubahan itu sebagai angka.

Kalimat penting:

A0 bukan pin daya. A0 adalah pin baca tegangan analog.

2.8 Konsep Ground Bersama dan Cara Membaca Pinout

Ground bersama

Ground bersama berarti semua perangkat dalam satu rangkaian memakai referensi nol volt yang sama.

Contoh:

```
WeMOS GND — LED GND
WeMOS GND — Button GND
WeMOS GND — Sensor GND
WeMOS GND — Power Supply GND
```

Jika dua perangkat saling bertukar sinyal, keduanya harus punya GND bersama.

Analogi sederhana:

Tegangan seperti tinggi badan. Tinggi badan hanya bermakna jika ada lantai sebagai acuan. Dalam rangkaian, **GND adalah lantainya**.

Cara membaca pinout WeMOS mini D1

Saat melihat pinout, cari dulu:

- **3V3.**
- **5V.**
- **GND.**
- **D0–D8.**
- **A0.**
- **TX/RX.**
- **SDA/SCL.**
- Pin yang perlu hati-hati saat *boot*.

Contoh pemetaan umum WeMOS mini D1:

Label Board	Fungsi Umum
3V3	Daya 3.3V
5V	Jalur daya 5V
GND	Ground
D0	GPIO digital
D1	GPIO / sering dipakai SCL
D2	GPIO / sering dipakai SDA
D3	GPIO, perlu hati-hati saat boot
D4	GPIO, sering terhubung LED internal
D5	GPIO digital
D6	GPIO digital
D7	GPIO digital
D8	GPIO, perlu hati-hati saat boot
A0	Input analog
TX	Serial transmit
RX	Serial receive

Catatan:

Pemetaan bisa berbeda antar board. Selalu cocokkan dengan pinout board yang ada di lab.

2.9 Praktik 1 — Menyalakan LED Eksternal dari WeMOS mini D1

Tujuan

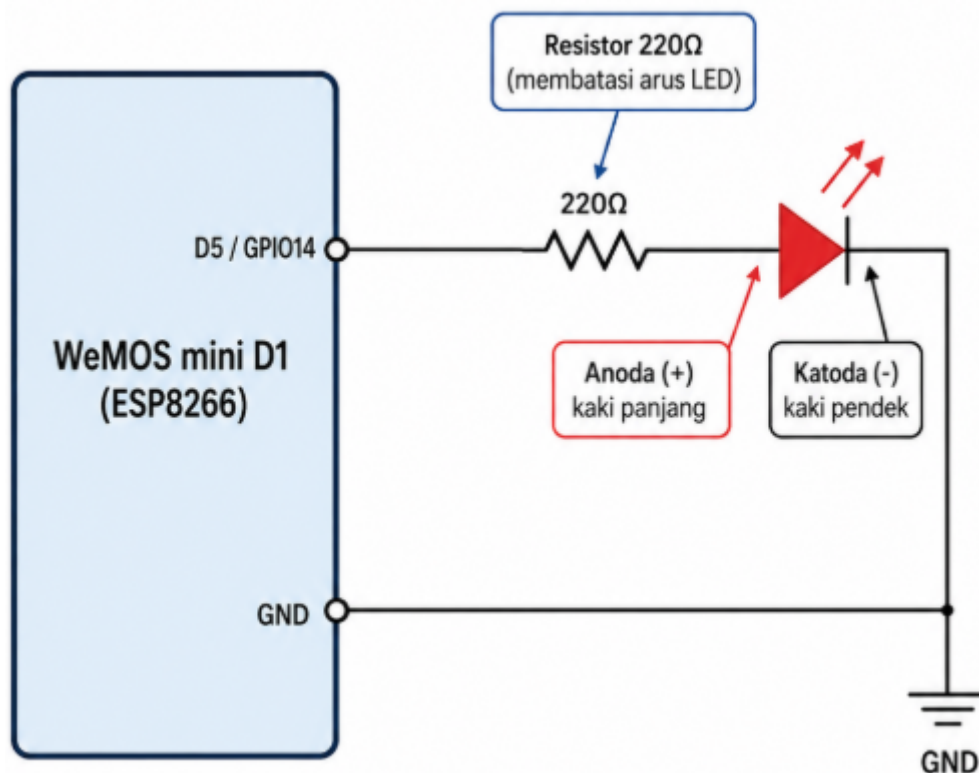
Siswa memahami hubungan antara pin digital, resistor, LED, dan GND.

Komponen

- WeMOS mini D1.
- Breadboard.
- LED.
- Resistor 220Ω atau 330Ω.
- Kabel jumper.
- Kabel USB data.

Konsep sambungan

D5 WeMOS → Resistor → Kaki panjang LED → Kaki pendek LED → GND



Gambar 2.3 LED eksternal harus memakai resistor agar arus tidak berlebihan dan pin WeMOS mini D1 tetap aman.

Tabel sambungan

Komponen	Sambungan
D5 WeMOS mini D1	Resistor
Resistor	Kaki panjang LED
Kaki pendek LED	GND

Kode contoh

```
int ledPin = D5;

void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
  digitalWrite(ledPin, HIGH);
  delay(1000);

  digitalWrite(ledPin, LOW);
  delay(1000);
}
```

Penjelasan kode singkat

- `int ledPin = D5;`
Membuat variabel bernama `ledPin` dan mengisinya dengan pin D5.
- `void setup()`
Bagian yang berjalan sekali saat board menyala.
- `pinMode(ledPin, OUTPUT);`
Mengatur D5 sebagai output.
- `void loop()`
Bagian yang berjalan berulang-ulang.
- `digitalWrite(ledPin, HIGH);`
Memberi sinyal HIGH agar LED menyala.
- `delay(1000);`
Menunggu 1000 milidetik atau 1 detik.
- `digitalWrite(ledPin, LOW);`
Memberi sinyal LOW agar LED mati.

Hasil yang diharapkan

LED menyala 1 detik, lalu mati 1 detik, berulang terus.

Jika LED tidak menyala

Periksa:

- LED terbalik.
- Resistor salah jalur.
- GND belum tersambung.
- Pin di kode tidak sama dengan rangkaian.
- Kabel USB bukan kabel data.
- Board belum dipilih dengan benar.

2.10 Praktik 2 — Membaca Tombol Sederhana

Tujuan

Siswa memahami input digital dan perubahan status tombol.

Komponen

- WeMOS mini D1.
- Push button.
- Breadboard.
- Kabel jumper.
- Resistor jika memakai rangkaian *pull-down* eksternal.

Konsep sederhana

Tombol adalah input digital. Nilainya hanya dua:

- **Ditekan.**
- **Tidak ditekan.**

Untuk praktik pemula, gunakan *internal pull-up* agar rangkaian lebih sederhana.

Sambungan

Komponen	Sambungan
Satu kaki button	D6
Kaki button lain	GND

Kode contoh

```
int buttonPin = D6;

void setup() {
  Serial.begin(9600);
  pinMode(buttonPin, INPUT_PULLUP);
}

void loop() {
  int statusTombol = digitalRead(buttonPin);

  if (statusTombol == LOW) {
    Serial.println("Tombol ditekan");
  } else {
    Serial.println("Tombol tidak ditekan");
  }
}
```

```
    }  
  
    delay(300);  
}
```

Penjelasan kode singkat

- `int buttonPin = D6;`
Menentukan tombol terhubung ke pin D6.
- `Serial.begin(9600);`
Mengaktifkan komunikasi serial dengan kecepatan 9600 bps.
- `pinMode(buttonPin, INPUT_PULLUP);`
Mengatur D6 sebagai input dengan resistor *pull-up* internal.
- `digitalRead(buttonPin);`
Membaca status tombol.
- `if (statusTombol == LOW)`
Pada mode *pull-up*, tombol yang ditekan biasanya terbaca LOW karena terhubung ke GND.
- `Serial.println(...)`
Menampilkan teks ke Serial Monitor.
- `delay(300);`
Memberi jeda agar teks tidak terlalu cepat.

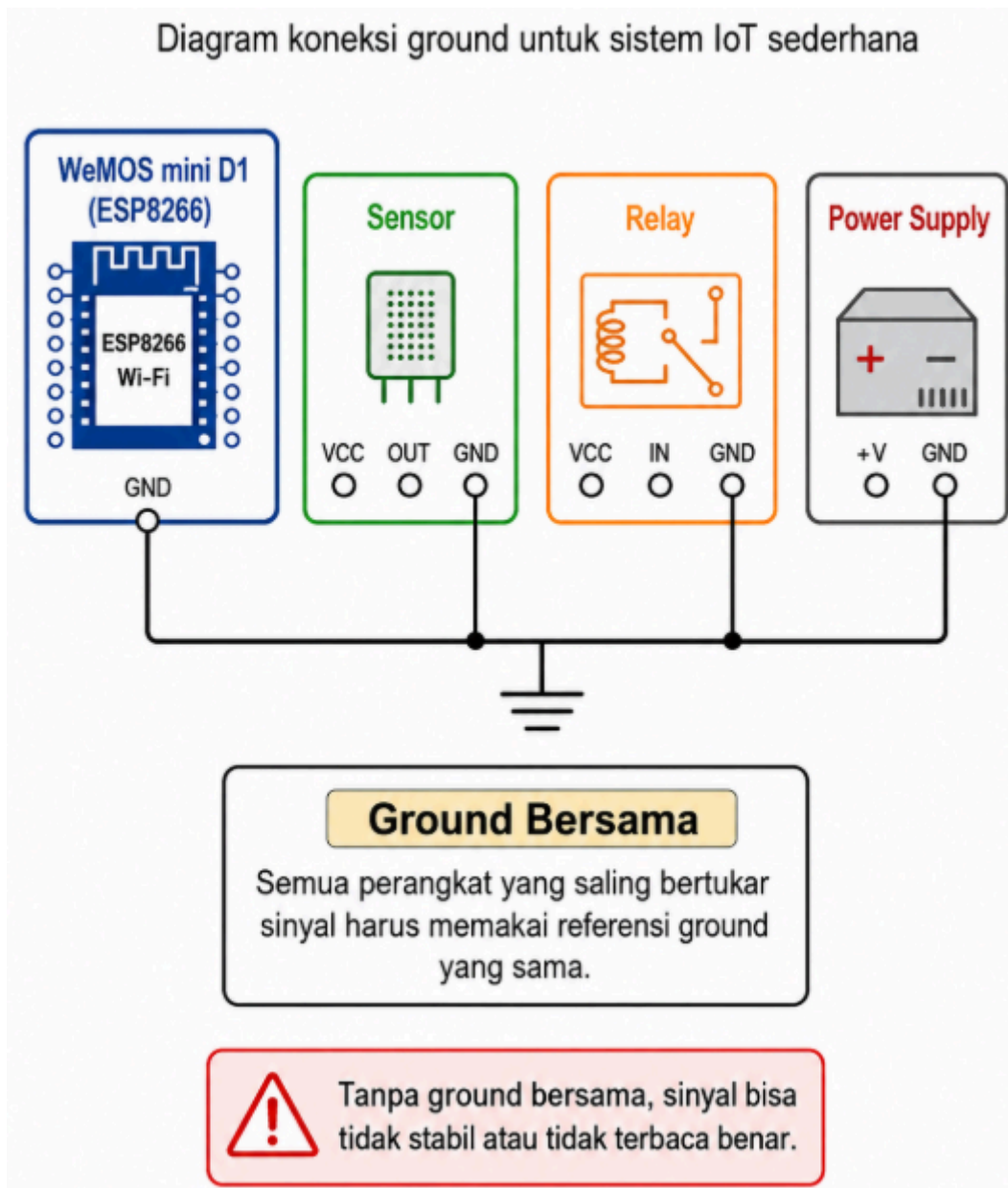
Hasil yang diharapkan

Saat tombol ditekan, Serial Monitor menampilkan:

Tombol ditekan

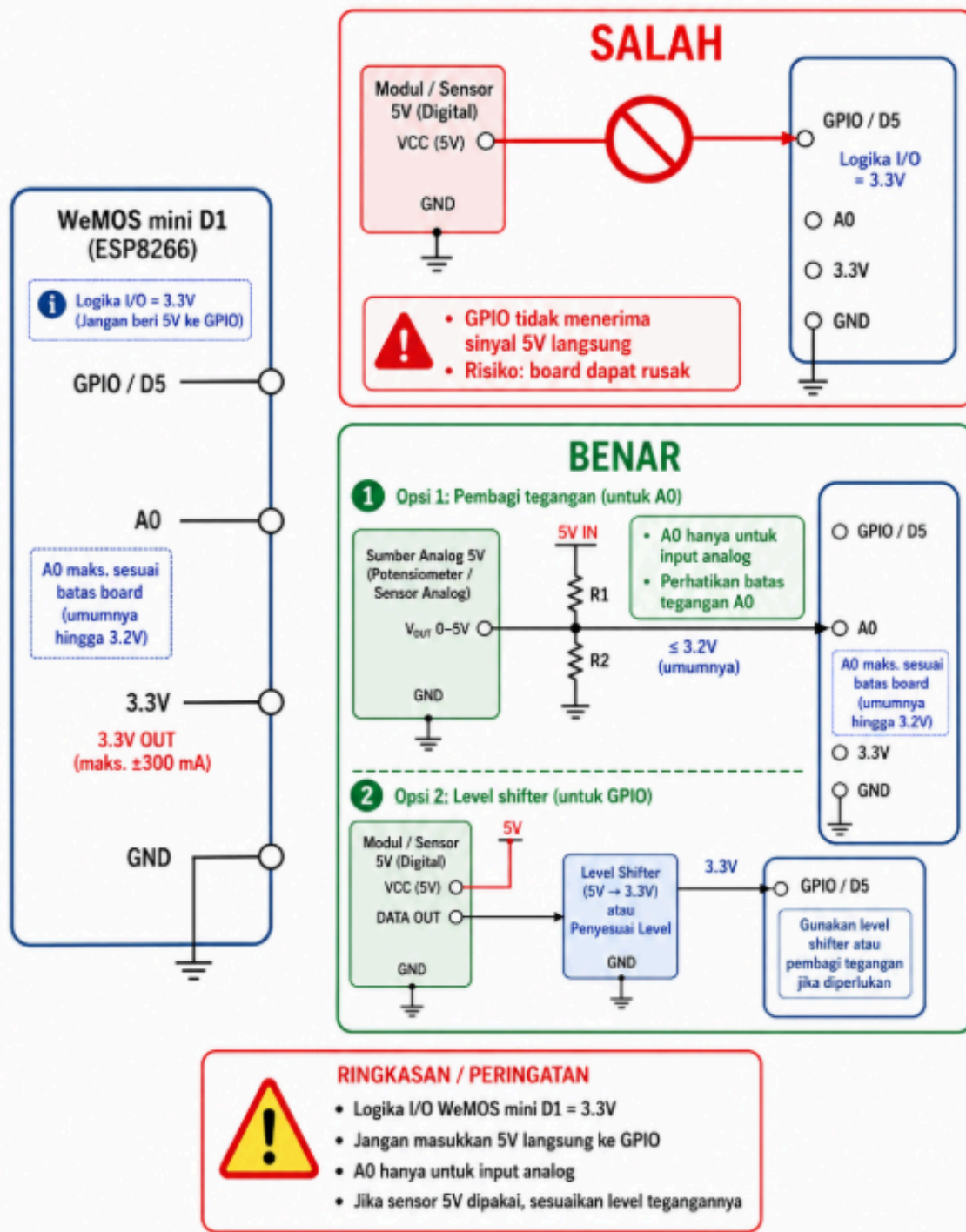
Saat tombol dilepas, Serial Monitor menampilkan:

Tombol tidak ditekan



Gambar 2.4 Semua perangkat yang saling bertukar sinyal harus memiliki ground bersama.

Diagram peringatan tegangan untuk WeMOS mini D1



Gambar 2.5 WeMOS mini D1 bekerja pada logika 3.3V; kesalahan tegangan dapat merusak board.

Checklist Sebelum Menyalakan Rangkaian

Gunakan checklist ini sebelum board diberi daya.

- Apakah VCC dan GND sudah benar?
- Apakah sensor bekerja di 3.3V atau 5V?
- Apakah pin I/O tidak menerima sinyal 5V langsung?
- Apakah A0 tidak menerima tegangan berlebih?
- Apakah LED sudah memakai resistor?
- Apakah tombol tersambung ke pin dan GND dengan benar?
- Apakah GND semua perangkat sudah tersambung?
- Apakah pin yang dipakai sama dengan kode?
- Apakah tidak ada kabel longgar?
- Apakah tidak ada hubungan pendek antara VCC dan GND?
- Apakah beban relay atau servo memakai catu daya yang sesuai?

Aturan terakhir:

Kalau ragu, jangan nyalakan. Periksa ulang.

Latihan Bab 2

1. Apa fungsi pin **3V3** pada WeMOS mini D1?
2. Mengapa **GND bersama** penting?
3. Apa bahaya memasukkan sinyal 5V ke pin I/O WeMOS mini D1?
4. Apa fungsi resistor pada rangkaian LED?
5. Mengapa A0 harus dipakai hati-hati?
6. Apa perbedaan D0–D8 dan A0?
7. Apa fungsi TX dan RX?
8. Mengapa relay dan servo tidak boleh langsung mengambil daya besar dari board?
9. Mengapa pinout harus dicek sebelum merangkai?
10. Apa manfaat praktik LED dan tombol sebelum masuk MQTT?

Tugas Praktik Bab 2

Buat dua rangkaian dasar:

1. **LED eksternal berkedip**
2. **Tombol terbaca di Serial Monitor**

Laporan singkat berisi:

- Foto rangkaian.
- Pin yang dipakai.
- Nilai resistor.
- Potongan kode.
- Hasil pengujian.
- Masalah yang ditemukan.
- Solusi yang dilakukan.

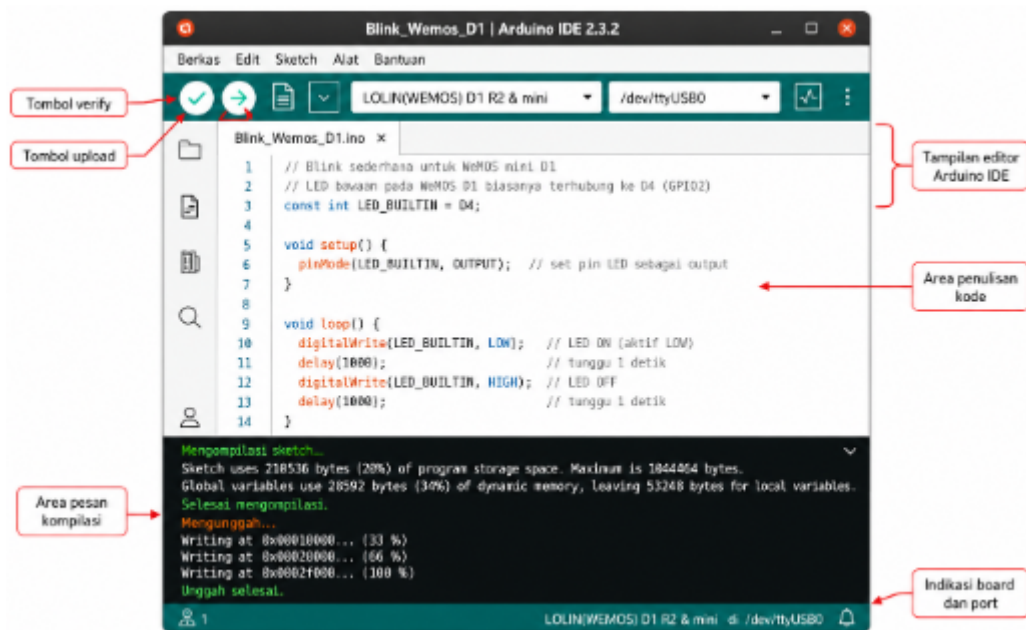
Target akhir Bab 2:

Siswa mampu menjelaskan **bentuk fisik WeMOS mini D1, fungsi pin penting, cara kerja breadboard, beda 3.3V dan 5V, batas A0, konsep ground bersama, serta membuat LED dan tombol bekerja dengan aman.**

BAB 3 — Instalasi Arduino IDE, ESP8266 Core, dan Program Pertama

3.1 Mengapa Arduino IDE Dipakai untuk Pemula

Arduino IDE dipakai karena sederhana, ringan, dan cocok untuk siswa SMK. Siswa dapat menulis kode, memilih board, memilih port, mengunggah program, dan melihat pesan debug dalam satu lingkungan kerja.



Gambar 3.1 Arduino IDE menjadi tempat menulis kode, memilih board, memilih port, mengunggah program, dan membaca pesan dari board.

Untuk WeMOS mini D1, Arduino IDE membutuhkan tambahan **ESP8266 Core**. Tanpa ESP8266 Core, Arduino IDE belum mengenal board berbasis ESP8266 seperti WeMOS mini D1.

ESP8266 Arduino Core menyediakan dukungan untuk pemrograman ESP8266 memakai gaya Arduino, termasuk *digital I/O*, *analog input*, *timing*, *serial*, Wi-Fi library, dan berbagai pustaka lain. Dokumentasi ESP8266 Arduino Core juga menyediakan bagian instalasi, Boards Manager, referensi *Digital IO*, *Analog input*, *Timing and delays*, *Serial*, dan library Wi-Fi. (arduino-esp8266.readthedocs.io)

Inti Bab 3:

Siswa harus bisa menyiapkan alat pemrograman sebelum membuat proyek IoT.

3.2 Persiapan Ubuntu Linux dan Kabel USB Data

Sebelum instalasi, siapkan:

- **Laptop / PC Ubuntu Linux.**
- **Arduino IDE.**
- **WeMOS mini D1.**
- **Kabel USB data.**
- **LED dan resistor** untuk praktik eksternal.
- **Koneksi internet** untuk mengunduh ESP8266 Core.

Kesalahan paling sering pada pemula adalah memakai kabel USB yang hanya bisa mengisi daya. Kabel seperti ini membuat board menyala, tetapi **port serial tidak muncul**.

Ciri kabel bermasalah:

- WeMOS mini D1 menyala.
- Lampu indikator hidup.
- Tetapi port tidak muncul di Arduino IDE.
- Upload selalu gagal.
- Serial Monitor tidak menampilkan apa-apa.

Aturan praktis:

Gunakan kabel USB data, bukan kabel charger saja.

WeMOS / LOLIN D1 mini secara resmi berbasis ESP8266EX, bekerja pada **3.3V**, memiliki **11 digital I/O**, **1 analog input maksimal 3.2V**, clock **80/160 MHz**, dan flash **4 MB**. Ini penting karena pilihan board dan batas tegangan harus sesuai dengan perangkat yang dipakai. ([Wemos](#))

3.3 Instalasi Arduino IDE di Ubuntu Linux

Arduino IDE dapat diinstal di Ubuntu Linux dari sumber resmi Arduino. Dokumentasi resmi Arduino IDE 2 menjelaskan bahwa untuk Linux, pengguna dapat mengunduh **AppImage 64-bit x86-64**, membuatnya dapat dieksekusi, lalu menjalankannya. ([Dokumentasi Arduino](#))

Langkah umum:

1. Buka halaman unduhan resmi Arduino.
2. Unduh Arduino IDE untuk Linux.
3. Jadikan file dapat dijalankan.
4. Jalankan Arduino IDE.
5. Pastikan tampilan editor terbuka normal.

Contoh perintah umum di Ubuntu Linux:

```
chmod +x arduino-ide_*.AppImage
./arduino-ide_*.AppImage
```

Penjelasan singkat:

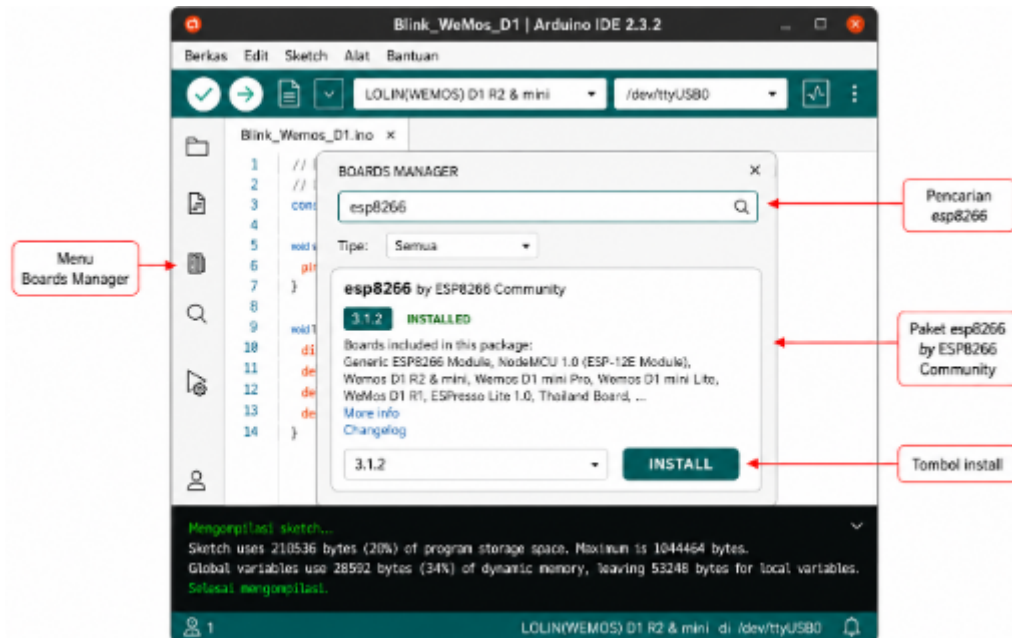
- `chmod +x` memberi izin agar file bisa dijalankan.
- `arduino-ide_*.AppImage` menunjuk file Arduino IDE AppImage.
- `./` menjalankan file dari folder saat ini.

Catatan editorial:

Perintah lengkap, variasi instalasi, dan solusi izin port serial sebaiknya diletakkan di lampiran agar Bab 3 tetap ringan untuk siswa.

3.4 Menambahkan ESP8266 Core melalui Boards Manager

Setelah Arduino IDE berjalan, langkah berikutnya adalah menambahkan dukungan board ESP8266.



Gambar 3.2 ESP8266 Core harus dipasang agar Arduino IDE mengenali board WeMos mini D1.

Dokumentasi ESP8266 Arduino Core menjelaskan langkah resmi: buka Preferences, masukkan URL

https://arduino.esp8266.com/stable/package_esp8266com_index.json ke **Additional Boards Manager URLs**, lalu buka Boards Manager, cari platform **esp8266**, pilih versi yang dibutuhkan, klik install, dan setelah selesai pilih board ESP8266 dari menu board. (arduino-esp8266.readthedocs.io)

Langkah ringkas:

1. Buka **Arduino IDE**.
2. Masuk ke **Preferences**.
3. Isi **Additional Boards Manager URLs** dengan:

```
https://arduino.esp8266.com/stable/package_esp8266com_in  
dex.json
```

4. Buka **Boards Manager**.
5. Cari:

```
esp8266
```

6. Install **esp8266 by ESP8266 Community**.
7. Tunggu proses selesai.
8. Restart Arduino IDE bila diperlukan.

Catatan penting:

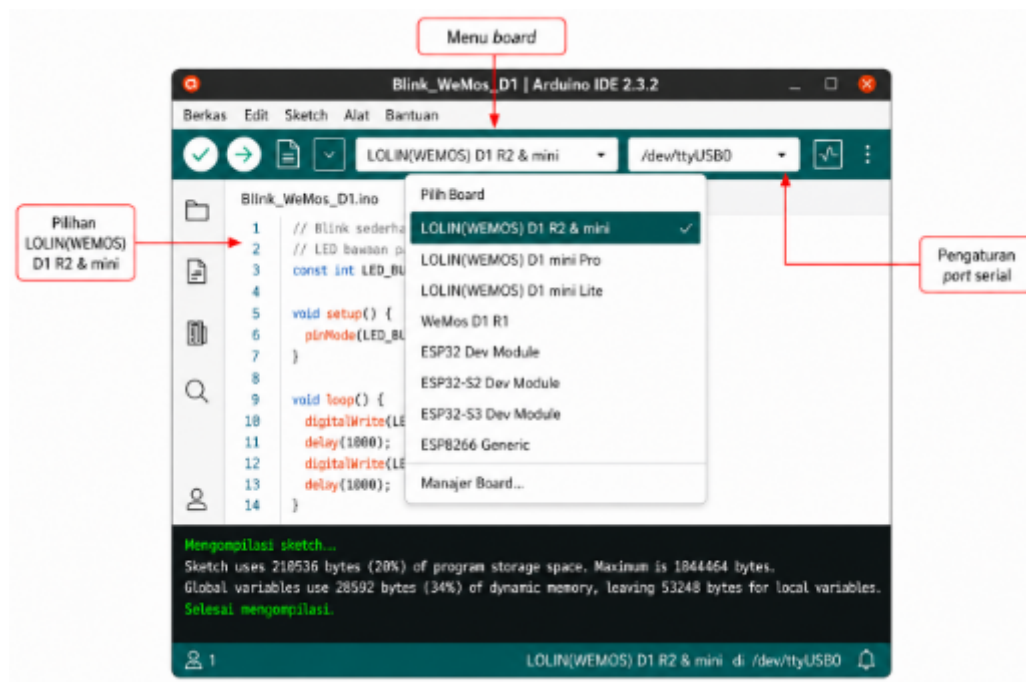
Gunakan URL resmi dari dokumentasi ESP8266 Arduino Core, bukan URL acak dari blog. Ini mengurangi risiko paket salah, versi tidak jelas, atau instruksi lama.

3.5 Memilih Board LOLIN(WEMOS) D1 R2 & mini dan Port Serial

Setelah ESP8266 Core terpasang, pilih board:

LOLIN(WEMOS) D1 R2 & mini

Board ini dipilih karena sesuai untuk keluarga WeMOS / LOLIN D1 mini yang dipakai dalam buku. Dokumentasi WeMOS juga menyatakan D1 mini kompatibel dengan Arduino, MicroPython, dan NodeMCU. ([Wemos](#))



Gambar 3.3 Board dan port yang benar menentukan keberhasilan proses upload ke WeMOS mini D1.

Pengaturan umum:

Item	Pilihan
Board	LOLIN(WEMOS) D1 R2 & mini
Upload Speed	115200 atau 921600
CPU Frequency	80 MHz atau 160 MHz
Flash Size	Sesuaikan board, umum 4 MB
Port	Port serial WeMOS mini D1

Untuk port serial di Ubuntu Linux, biasanya muncul sebagai:

```
/dev/ttyUSB0
```

atau:

```
/dev/ttyACM0
```

Jika port tidak muncul, penyebab umum:

- Kabel hanya untuk *charging*.
- Driver USB serial belum cocok.
- Board belum terdeteksi.
- Izin akses port serial belum benar.
- Board belum tersambung rapat.

Kalimat penting:

Board benar + port benar + kabel data benar = peluang upload berhasil jauh lebih besar.

3.6 Struktur Program: setup() dan loop()

Program Arduino memiliki dua bagian utama:

```
void setup() {  
    // berjalan sekali saat board menyala  
}  
  
void loop() {  
    // berjalan berulang-ulang selama board hidup  
}
```

setup()

Bagian `setup()` berjalan satu kali saat board menyala atau di-reset.

Biasanya dipakai untuk:

- Mengatur pin sebagai input atau output.
- Mengaktifkan Serial Monitor.
- Menyiapkan Wi-Fi.
- Menyiapkan MQTT.
- Menyiapkan sensor.

Contoh:

```
void setup() {  
    pinMode(D5, OUTPUT);  
}
```

loop()

Bagian `loop()` berjalan terus-menerus.

Biasanya dipakai untuk:

- Membaca sensor.
- Menyalakan atau mematikan LED.
- Mengirim data.
- Mengecek koneksi.
- Menunggu perintah.

Contoh:

```
void loop() {  
    digitalWrite(D5, HIGH);  
    delay(1000);  
    digitalWrite(D5, LOW);  
    delay(1000);  
}
```

Pola pikir penting:

`setup()` menyiapkan. `loop()` menjalankan pekerjaan utama berulang-ulang.

3.7 Praktik 1 — Blink LED Internal

Tujuan

Siswa memastikan Arduino IDE, ESP8266 Core, board, port, kabel, dan proses upload sudah benar.

Konsep

Banyak board WeMOS mini D1 memiliki LED internal. Pada beberapa board, LED internal terhubung ke **D4 / GPIO2**. Dikenali di Code sebagai `LED_BUILTIN`. Bagi yang ingin memasang LED diluar bisa menggunakan **D5 / GPIO14** seperti tampak pada Gambar. Dalam Code, **D5** dikenali sebagai `D5`. Namun perilakunya bisa aktif rendah, artinya LED menyala saat pin diberi nilai **LOW**.

Kode

```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);

  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000);
}
```

Penjelasan Kode

- `pinMode(LED_BUILTIN, OUTPUT);`
Mengatur LED internal sebagai output.
- `digitalWrite(LED_BUILTIN, LOW);`
Pada banyak board ESP8266, ini menyalakan LED internal.
- `delay(1000);`
Menunggu 1 detik.
- `digitalWrite(LED_BUILTIN, HIGH);`
Mematikan LED internal pada banyak board.

Hasil yang diharapkan:

LED internal berkedip setiap 1 detik.

Catatan kritis:

Jika LED internal tidak berkedip, belum tentu board rusak. Bisa saja board memakai pin LED berbeda atau LED internal aktif dengan logika terbalik.

3.8 Praktik 2 — Blink LED Eksternal

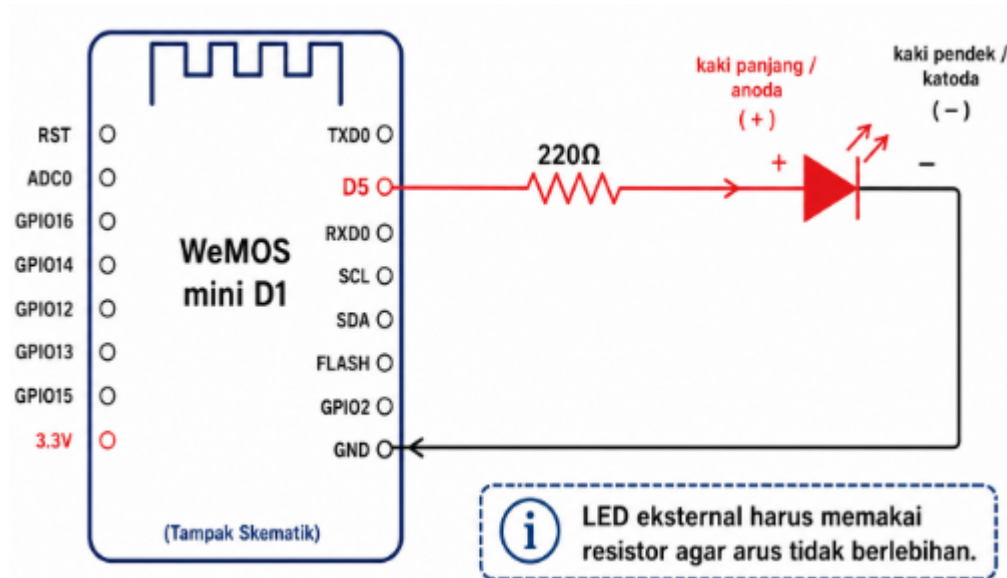
Tujuan

Siswa memahami output digital, resistor, LED, dan GND.

Rangkaian

Gunakan pola:

D5 → Resistor 220Ω → kaki panjang LED → kaki pendek LED
→ GND



Gambar 3.4 LED eksternal harus memakai resistor agar arus tidak berlebihan dan pin WeMos mini D1 tetap aman.

Aturan keras:

LED eksternal harus memakai resistor agar arus tidak berlebihan.

Komponen

- WeMos mini D1.
- LED.
- Resistor 220Ω atau 330Ω.
- Breadboard.
- Kabel jumper.
- Kabel USB data.

Kode

```
int ledPin = D5;

void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
  digitalWrite(ledPin, HIGH);
  delay(1000);

  digitalWrite(ledPin, LOW);
  delay(1000);
}
```

Penjelasan Kode

- `int ledPin = D5;`
Membuat variabel `ledPin` berisi pin D5.
- `pinMode(ledPin, OUTPUT);`
Menjadikan D5 sebagai output.
- `digitalWrite(ledPin, HIGH);`
Memberi logika HIGH ke D5 sehingga LED menyala.
- `digitalWrite(ledPin, LOW);`
Memberi logika LOW ke D5 sehingga LED mati.

Hasil yang diharapkan:

LED eksternal menyala 1 detik dan mati 1 detik secara berulang.

3.9 Praktik 3 — Menampilkan Teks di Serial Monitor

Tujuan

Siswa memahami bahwa *Serial Monitor* adalah alat penting untuk melihat pesan dari board.

Serial sangat berguna untuk:

- Mengecek board hidup.
- Melihat nilai sensor.
- Melihat status Wi-Fi.
- Melihat status MQTT.
- Menampilkan pesan error.
- Menguji komunikasi dengan komputer.

Kode

```
void setup() {  
  Serial.begin(9600);  
  Serial.println("WeMOS mini D1 siap belajar IoT");  
}  
  
void loop() {  
  Serial.println("Program berjalan...");  
  delay(1000);  
}
```

Penjelasan Kode

- `Serial.begin(9600);`
Mengaktifkan komunikasi serial pada kecepatan 9600 bps.
- `Serial.println("WeMOS mini D1 siap belajar IoT");`
Mengirim teks ke Serial Monitor.
- `Serial.println("Program berjalan...");`
Mengirim teks berulang setiap 1 detik.

Hasil yang diharapkan

Serial Monitor menampilkan:

```
WeMOS mini D1 siap belajar IoT  
Program berjalan...  
Program berjalan...  
Program berjalan...
```

Catatan penting:

Kecepatan di Serial Monitor harus sama dengan kode. Jika kode memakai 9600, Serial Monitor juga pilih 9600.

3.10 Masalah Umum dan Troubleshooting Upload

1. Port tidak muncul

Penyebab umum:

- Kabel USB hanya untuk *charging*.
- Board belum tersambung baik.
- Port belum diberi izin.
- Chip USB-to-serial belum dikenali.

Solusi:

- Ganti kabel USB data.
- Cabut dan pasang ulang board.
- Cek port di Arduino IDE.
- Cek perangkat serial di Ubuntu.

Contoh cek port:

```
ls /dev/ttyUSB*  
ls /dev/ttyACM*
```

2. Kabel USB hanya untuk charging

Gejala:

- Board menyala.
- Tidak ada port.
- Upload gagal.

Solusi:

- Gunakan kabel data.
- Uji kabel dengan perangkat lain.
- Tandai kabel data khusus untuk praktik.

3. Driver USB serial

Beberapa board memakai chip USB-to-serial seperti CH340 atau CP210x. Dokumentasi LOLIN D1 mini menyediakan rujukan driver CH340 sebagai bagian dari dokumentasi board. ([Wemos](#))

Di Ubuntu Linux, banyak chip sudah dikenali otomatis, tetapi tetap perlu dicek jika port tidak muncul.

4. Board salah pilih

Gejala:

- Upload gagal.
- Program tidak jalan normal.
- Pin tidak sesuai.

Solusi:

- Pilih **LOLIN(WEMOS) D1 R2 & mini**.
- Pastikan ESP8266 Core sudah terpasang.
- Cek kembali menu board.

5. Upload gagal

Penyebab umum:

- Port salah.
- Board salah.
- Kabel bermasalah.
- Program masih error.
- Board masuk kondisi boot tidak normal.
- Pin tertentu mengganggu proses boot.

Solusi:

- Cek ulang board dan port.
- Lepas rangkaian eksternal dulu.
- Upload program paling sederhana.
- Hindari beban pada pin boot saat upload.
- Tekan tombol reset jika diperlukan.

Kalimat penting:

Saat upload gagal, jangan langsung menyalahkan kode. Periksa kabel, port, board, dan rangkaian.

Latihan Bab 3

1. Mengapa Arduino IDE perlu ditambah ESP8266 Core?
2. Apa fungsi **Boards Manager**?
3. Mengapa board harus dipilih **LOLIN(WEMOS) D1 R2 & mini**?
4. Apa akibat jika port serial salah?
5. Apa perbedaan `setup()` dan `loop()`?
6. Mengapa LED eksternal harus memakai resistor?
7. Apa fungsi `Serial.begin(9600)`?
8. Mengapa kabel USB charger-only sering membuat upload gagal?
9. Apa yang harus dicek pertama kali saat port tidak muncul?
10. Mengapa rangkaian eksternal sebaiknya dilepas saat upload bermasalah?

Tugas Praktik Bab 3

Buat tiga program dasar:

1. **Blink LED internal.**
2. **Blink LED eksternal di D5.**
3. **Serial Monitor menampilkan teks:**
WeMOS mini D1 siap belajar IoT

Laporan singkat berisi:

- Screenshot Arduino IDE.
- Board yang dipilih.
- Port serial yang dipakai.
- Kode program.
- Hasil upload.
- Foto LED berkedip.
- Screenshot Serial Monitor.
- Masalah yang muncul.
- Solusi yang dilakukan.

Target akhir Bab 3:

Siswa mampu menginstal Arduino IDE di Ubuntu Linux, memasang ESP8266 Core, memilih board dan port dengan benar, mengunggah program pertama, menyalakan LED, dan membaca pesan dari Serial Monitor.

BAB 4 — Input dan Output Dasar: Button, LED, dan Potensio

4.1 Apa Itu Input dan Output

Dalam sistem IoT, **input** adalah data yang masuk ke board. **Output** adalah aksi atau sinyal yang keluar dari board.

Contoh input:

- **Button** ditekan atau dilepas.
- **Potensiometer** diputar.
- Sensor suhu membaca nilai.
- Sensor cahaya membaca terang atau gelap.

Contoh output:

- **LED** menyala atau mati.
- Buzzer berbunyi.
- Relay aktif.
- Servo bergerak.
- Data dikirim ke Serial Monitor.

Pada Bab 4, siswa belajar tiga hal dasar:

- **Output digital** melalui LED.
- **Input digital** melalui button.
- **Input analog** melalui potensiometer.

WeMOS mini D1 cocok untuk praktik ini karena memiliki **11 digital I/O** dan **1 analog input**. Dokumentasi resmi LOLIN menyebut D1 mini bekerja pada **3.3V**, memiliki **11 digital I/O**, dan **1 analog input dengan batas 3.2V maksimum**. ([Wemos](#))

Kalimat penting:

Input membuat board membaca dunia. **Output** membuat board memberi aksi.

4.2 LED sebagai Output Digital

LED adalah komponen paling sederhana untuk memahami output digital.

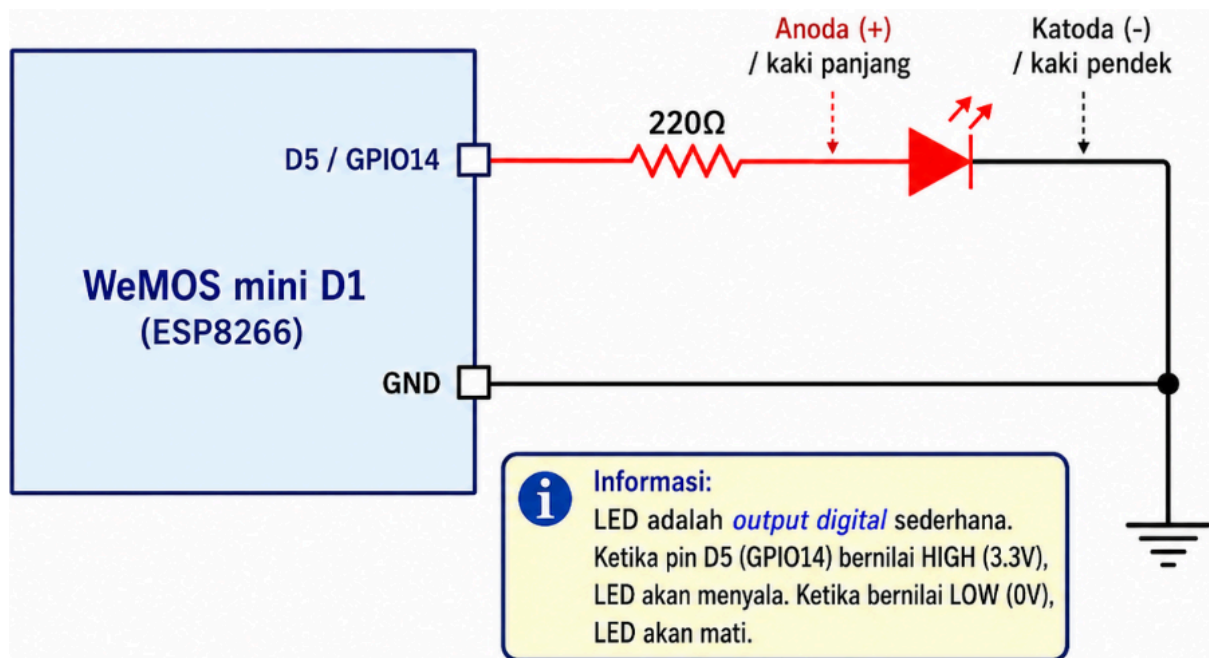
Output digital hanya memiliki dua keadaan:

- **HIGH** — sinyal aktif.
- **LOW** — sinyal tidak aktif.

Pada rangkaian LED eksternal, pola yang digunakan:

D5 → Resistor 220Ω → LED → GND

Resistor wajib digunakan agar arus tidak berlebihan. Tanpa resistor, LED bisa rusak dan pin WeMOS mini D1 dapat terbebani.



Gambar 4.1 LED adalah output digital paling sederhana untuk melihat hasil program.

Komponen:

- WeMOS mini D1.
- LED.
- Resistor 220Ω atau 330Ω.
- Breadboard.
- Kabel jumper.

Kode LED ON/OFF:

```
int ledPin = D5;

void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
  digitalWrite(ledPin, HIGH);
  delay(1000);

  digitalWrite(ledPin, LOW);
  delay(1000);
}
```

Penjelasan singkat:

- `ledPin` menyimpan pin LED.
- `pinMode()` mengatur pin sebagai output.
- `digitalWrite()` mengubah pin menjadi HIGH atau LOW.
- `delay(1000)` memberi jeda 1 detik.

Target pemahaman:

Siswa paham bahwa LED adalah indikator output paling mudah untuk melihat kerja program.

4.3 Button sebagai Input Digital

Button adalah input digital. Nilainya sederhana:

- Ditekan.
- Tidak ditekan.

Dalam program, button dibaca memakai `digitalRead()`. Fungsi `digitalRead()` membaca nilai dari pin digital dan mengembalikan nilai HIGH atau LOW. Dokumentasi Arduino menjelaskan `digitalRead()` sebagai fungsi untuk membaca nilai dari pin digital tertentu. ([Arduino](#))

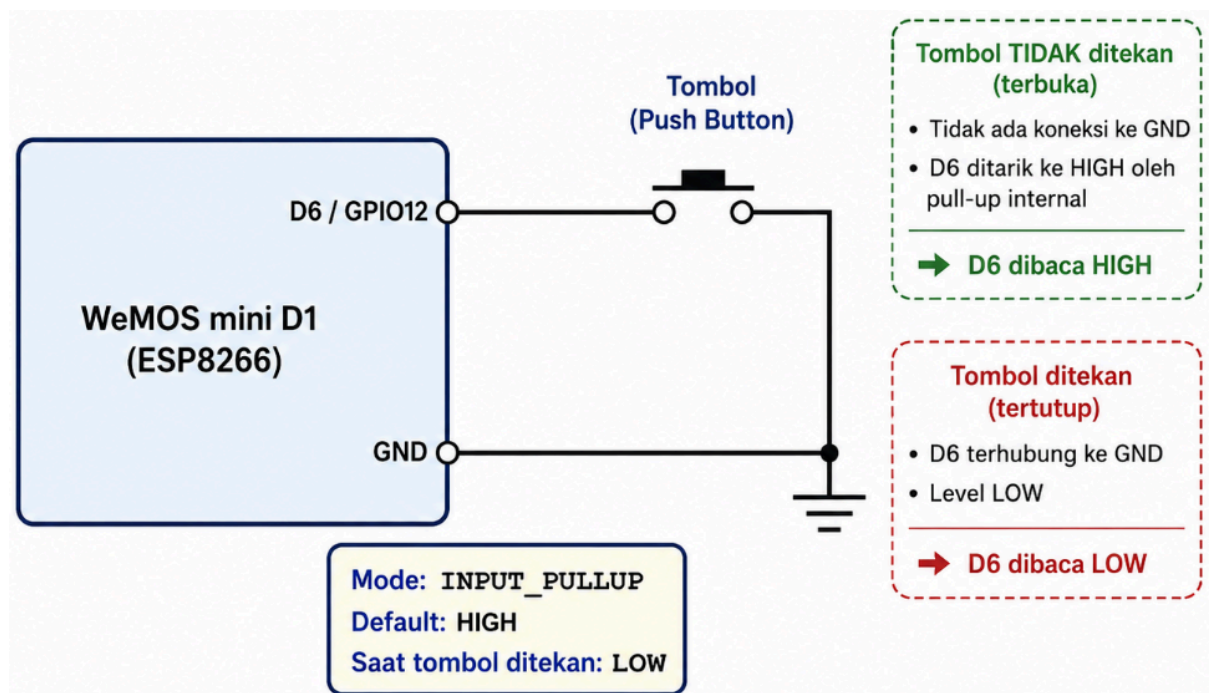
Untuk praktik pemula, lebih aman memakai mode:

```
INPUT_PULLUP
```

Dengan mode ini, resistor *pull-up* internal digunakan. Rangkaian menjadi lebih sederhana.

Rangkaian sederhana:

D6 → Button → GND



Gambar 4.2 Button menjadi input digital yang aman dan sederhana saat memakai mode `INPUT_PULLUP`.

Kode membaca button:

```
int buttonPin = D6;

void setup() {
  Serial.begin(9600);
  pinMode(buttonPin, INPUT_PULLUP);
}

void loop() {
  int statusTombol = digitalRead(buttonPin);

  if (statusTombol == LOW) {
    Serial.println("Tombol ditekan");
  } else {
    Serial.println("Tombol tidak ditekan");
  }

  delay(300);
}
```

Catatan penting:

Pada **INPUT_PULLUP**, tombol yang ditekan biasanya terbaca **LOW**, bukan HIGH.

Kalimat penting:

Button sederhana, tetapi logikanya harus dipahami agar tidak salah membaca status.

4.4 Konsep Pull-Up dan Pull-Down

Pin input tidak boleh dibiarkan mengambang. Jika pin input tidak tersambung ke keadaan jelas, nilainya bisa berubah-ubah. Ini disebut *floating input*.

Ada dua pendekatan umum:

1. Pull-up

Pin dibuat default **HIGH**. Saat tombol ditekan ke GND, nilai menjadi **LOW**.

Pola:

```
Default: HIGH  
Ditekan: LOW
```

Mode praktis:

```
pinMode(buttonPin, INPUT_PULLUP);
```

Kelebihan:

- Rangkaian lebih sederhana.
- Tidak perlu resistor eksternal.
- Cocok untuk pemula.

2. Pull-down

Pin dibuat default **LOW**. Saat tombol ditekan ke tegangan positif, nilai menjadi **HIGH**.

Pola:

```
Default: LOW  
Ditekan: HIGH
```

Kelemahan untuk pemula:

- Biasanya butuh resistor eksternal.
- Lebih mudah salah sambung.

Rekomendasi Bab 4:

Gunakan **INPUT_PULLUP** untuk praktik awal button.

4.5 Potensiometer sebagai Input Analog

Potensiometer adalah komponen yang dapat diputar untuk menghasilkan perubahan tegangan.

Potensiometer memiliki tiga kaki:

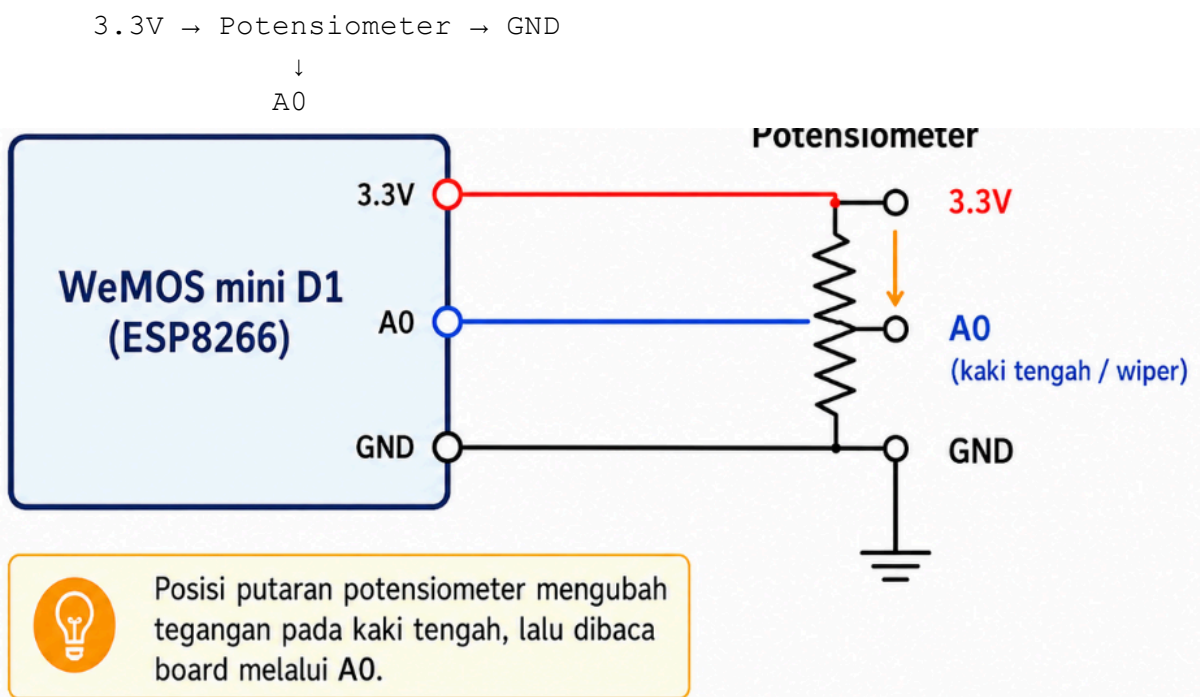
- Kaki kiri ke **3.3V**.
- Kaki tengah ke **A0**.
- Kaki kanan ke **GND**.

Saat diputar, tegangan pada kaki tengah berubah. WeMOS mini D1 membaca perubahan itu melalui pin **A0**.

Potensiometer cocok untuk menjelaskan:

- Input analog.
- Perubahan nilai bertahap.
- Konsep pembagi tegangan.
- Hubungan tegangan dan angka digital.
- Dasar kontrol LED dengan nilai sensor.

Pola rangkaian:



Gambar 4.3 Potensiometer mengubah posisi putaran menjadi tegangan analog yang dibaca melalui A0.

Catatan keras:

A0 hanya untuk input analog. Jangan dipakai sebagai output LED.

4.6 Pin A0 pada WeMOS mini D1

WeMOS mini D1 hanya memiliki **1 pin analog**, yaitu **A0**.

Dokumentasi LOLIN D1 mini menyebut:

- **Analog Input Pins: 1**
- **A0 maksimum 3.2V**
- **Operating Voltage: 3.3V** ([Wemos](#))

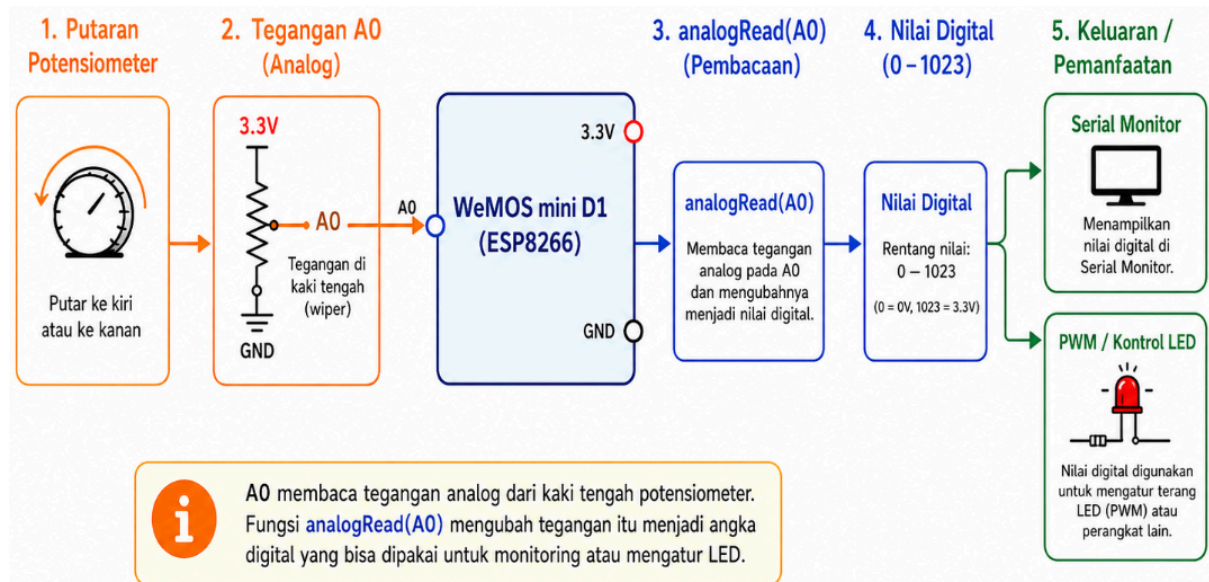
Artinya, siswa harus hati-hati. Jangan memasukkan tegangan sembarangan ke A0.

Aturan aman A0:

- Gunakan potensiometer dari **3.3V ke GND**.
- Ambil sinyal tengah potensiometer ke **A0**.
- Jangan masukkan tegangan di atas batas board.
- Jika memakai sensor 5V, gunakan pembagi tegangan.
- Pastikan GND sensor dan WeMOS tersambung.

Kalimat penting:

A0 membaca tegangan, bukan menerima perintah digital.



Gambar 4.4 A0 mengubah tegangan analog menjadi angka digital yang dapat dipakai untuk monitoring atau kontrol LED.

4.7 Membaca Nilai Analog dengan analogRead()

Fungsi `analogRead()` digunakan untuk membaca nilai analog dari A0.

Contoh:

```
int nilaiPot = analogRead(A0);
```

Nilai yang terbaca biasanya berupa angka digital. Angka ini mewakili tegangan yang masuk ke A0.

Kode dasar membaca A0:

```
void setup() {  
    Serial.begin(9600);  
}  
  
void loop() {  
    int nilaiPot = analogRead(A0);  
  
    Serial.print("Nilai potensiometer: ");  
    Serial.println(nilaiPot);  
  
    delay(300);  
}
```

Penjelasan singkat:

- `analogRead(A0)` membaca tegangan di A0.
- Nilainya disimpan ke `nilaiPot`.
- `Serial.print()` menulis teks tanpa pindah baris.
- `Serial.println()` menulis nilai lalu pindah baris.
- `delay(300)` memberi jeda pembacaan.

Target pemahaman:

Siswa melihat bahwa gerakan fisik potensiometer berubah menjadi angka di Serial Monitor.

4.8 Menampilkan Data Potensiometer ke Serial Monitor

Serial Monitor adalah alat penting untuk melihat data dari board.

Untuk potensiometer, Serial Monitor membantu siswa melihat hubungan antara:

- posisi putaran potensiometer,
- tegangan pada A0,
- nilai digital hasil pembacaan.

Contoh tampilan:

```
Nilai potensiometer: 0
Nilai potensiometer: 178
Nilai potensiometer: 532
Nilai potensiometer: 847
Nilai potensiometer: 1023
```

Nilai akhir dapat berbeda tergantung board, pustaka, dan rentang ADC. Yang penting bukan angka mutlaknyanya, tetapi pola perubahan saat potensiometer diputar.

Kode dengan status sederhana:

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  int nilaiPot = analogRead(A0);

  Serial.print("Nilai A0 = ");
  Serial.print(nilaiPot);

  if (nilaiPot < 300) {
    Serial.println(" | Rendah");
  } else if (nilaiPot < 700) {
    Serial.println(" | Sedang");
  } else {
    Serial.println(" | Tinggi");
  }

  delay(300);
}
```

Kalimat penting:

Serial Monitor membuat data yang tidak terlihat menjadi terlihat.

4.9 Mengendalikan LED dengan Nilai Analog dan PWM

Setelah siswa bisa membaca potensiometer, nilai itu dapat dipakai untuk mengendalikan LED.

Teknik yang dipakai adalah **PWM** atau *Pulse Width Modulation*. PWM membuat LED tampak lebih redup atau lebih terang dengan mengatur pulsa digital cepat.

Pada ESP8266 Arduino Core, dukungan ESP8266 memungkinkan penggunaan fungsi Arduino dan pustaka familiar untuk menjalankan program langsung pada ESP8266. ([GitHub](#))

Konsep sederhana:

- Potensiometer diputar kecil → LED redup.
- Potensiometer diputar besar → LED terang.

Rangkaian:

Potensiometer → A0
LED eksternal → D5

Kode PWM sederhana:

```
int ledPin = D5;

void setup() {
  Serial.begin(9600);
  pinMode(ledPin, OUTPUT);
}

void loop() {
  int nilaiPot = analogRead(A0);

  int terangLED = map(nilaiPot, 0, 1023, 0, 255);

  analogWrite(ledPin, terangLED);

  Serial.print("A0 = ");
  Serial.print(nilaiPot);
  Serial.print(" | PWM = ");
  Serial.println(terangLED);

  delay(100);
}
```

Penjelasan singkat:

- `analogRead(A0)` membaca potensiometer.
- `map()` mengubah rentang nilai A0 menjadi nilai PWM.
- `analogWrite()` mengatur kecerahan LED.
- Serial Monitor menampilkan nilai A0 dan PWM.

Catatan:

Rentang PWM pada ESP8266 dapat berbeda tergantung konfigurasi. Untuk pembelajaran awal, contoh 0–255 mudah dipahami siswa.

4.10 Kesalahan Umum Saat Membaca Tombol dan Potensio

Kesalahan pada button

Kesalahan umum:

- Tombol salah kaki.
- Pin input dibiarkan *floating*.
- Tidak memakai `INPUT_PULLUP`.
- Logika ditekan dianggap HIGH padahal pada `INPUT_PULLUP` terbaca LOW.
- GND tidak tersambung.
- Kabel longgar.

Solusi:

- Gunakan `INPUT_PULLUP`.
- Sambungkan tombol antara pin dan GND.
- Uji dengan Serial Monitor.
- Tampilkan status tombol secara jelas.

Kesalahan pada potensiometer

Kesalahan umum:

- Kaki tengah potensiometer tidak ke A0.
- 3.3V dan GND tertukar.
- A0 menerima tegangan terlalu tinggi.
- GND tidak tersambung.
- Nilai naik turun karena kabel longgar.
- Sensor 5V langsung masuk A0 tanpa penyesuaian tegangan.

Solusi:

- Pastikan kaki tengah ke A0.
- Gunakan 3.3V, bukan 5V.
- Cek batas A0.
- Gunakan pembagi tegangan jika perlu.
- Gunakan Serial Monitor untuk melihat nilai.

Aturan keras:

Jika nilai input tidak masuk akal, cek rangkaian dulu sebelum menyalahkan kode.

4.11 Praktik Gabungan: LED, Button, dan Potensio

Bab ini sebaiknya ditutup dengan praktik gabungan sederhana.

Tujuan

Siswa menggabungkan input digital, input analog, dan output digital.

Fungsi sistem

- Button menyalakan atau mematikan LED.
- Potensiometer mengatur kecepatan kedip LED.
- Serial Monitor menampilkan status button dan nilai A0.

Komponen

- WeMOS mini D1.
- LED.
- Resistor 220Ω / 330Ω .
- Button.
- Potensiometer.
- Breadboard.
- Kabel jumper.
- Kabel USB data.

Rangkaian ringkas

LED:

D5 → Resistor 220Ω → LED → GND

Button:

D6 → Button → GND

Potensiometer:

3.3V → kaki luar potensiometer

A0 → kaki tengah potensiometer

GND → kaki luar potensiometer

Kode gabungan

```
int ledPin = D5;
int buttonPin = D6;

void setup() {
  Serial.begin(9600);

  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT_PULLUP);
}

void loop() {
  int statusTombol = digitalRead(buttonPin);
  int nilaiPot = analogRead(A0);

  int jeda = map(nilaiPot, 0, 1023, 100, 1000);

  Serial.print("Tombol = ");
  Serial.print(statusTombol == LOW ? "DITEKAN" : "LEPAS");
  Serial.print(" | A0 = ");
  Serial.print(nilaiPot);
  Serial.print(" | Jeda = ");
  Serial.println(jeda);

  if (statusTombol == LOW) {
    digitalWrite(ledPin, HIGH);
    delay(jeda);

    digitalWrite(ledPin, LOW);
    delay(jeda);
  } else {
    digitalWrite(ledPin, LOW);
  }
}
```

Hasil yang diharapkan

- Saat tombol ditekan, LED berkedip.
- Kecepatan kedip berubah saat potensiometer diputar.
- Saat tombol dilepas, LED mati.
- Serial Monitor menampilkan status tombol, nilai A0, dan jeda.

Target akhir praktik:

Siswa memahami hubungan input digital, input analog, output digital, dan perubahan program.

4.12 Latihan, Tugas, dan Checklist Hasil

Latihan Bab 4

1. Apa perbedaan input dan output?
2. Mengapa LED disebut output digital?
3. Mengapa button disebut input digital?
4. Apa arti HIGH dan LOW?
5. Mengapa pin input tidak boleh *floating*?
6. Apa fungsi `INPUT_PULLUP`?
7. Mengapa tombol yang ditekan bisa terbaca LOW?
8. Apa fungsi A0 pada WeMOS mini D1?
9. Mengapa potensiometer cocok untuk belajar input analog?
10. Mengapa nilai A0 perlu ditampilkan di Serial Monitor?
11. Apa fungsi `map()`?
12. Mengapa PWM dapat mengatur terang LED?

Tugas Praktik Bab 4

Buat tiga program:

1. **Button menyalakan LED.**
2. **Potensiometer tampil di Serial Monitor.**
3. **Potensiometer mengatur kecerahan LED.**

Laporan singkat berisi:

- Foto rangkaian.
- Pin yang digunakan.
- Kode program.
- Screenshot Serial Monitor.
- Hasil uji.
- Masalah yang muncul.
- Solusi yang dilakukan.

Checklist Hasil

Siswa dinyatakan berhasil jika mampu:

- Menjelaskan beda **input** dan **output**.
- Menyalakan LED dari pin D5.
- Membaca button dari pin D6.
- Menjelaskan `INPUT_PULLUP`.
- Membaca potensiometer dari A0.
- Menampilkan nilai A0 di Serial Monitor.
- Menggunakan nilai analog untuk mengatur LED.
- Menjelaskan kesalahan umum dan cara memperbaikinya.

Catatan Akhir Bab 4

Bab 4 harus sangat kuat karena menjadi jembatan dari praktek listrik dasar ke IoT. Jangan terlalu cepat masuk ke Wi-Fi dan MQTT. Siswa harus lebih dulu nyaman dengan **LED**, **button**, **potensiometer**, **Serial Monitor**, dan **logika program sederhana**.

Penutup Bab 4:

Jika siswa sudah bisa membaca button, membaca potensiometer, dan mengendalikan LED, maka siswa sudah memahami inti awal IoT: membaca kondisi, mengolah nilai, dan memberi aksi.

BAB 5 — Jam Digital Sederhana dengan Serial Port ke Komputer

5.1 Mengapa Komunikasi Serial Penting Sebelum IoT

IoT tidak selalu langsung dimulai dari internet. Dalam praktek mikrokontroler, komunikasi pertama yang paling penting adalah **serial**.

Melalui serial, WeMOS mini D1 dapat mengirim pesan ke komputer. Siswa bisa melihat apakah program berjalan, apakah nilai variabel berubah, dan apakah perintah dari komputer diterima dengan benar.

Sebelum masuk ke MQTT, siswa harus memahami:

- Bagaimana board mengirim teks ke komputer.
- Bagaimana komputer membaca pesan dari board.
- Bagaimana komputer mengirim perintah ke board.
- Bagaimana board merespons perintah tersebut.
- Bagaimana data kecil diformat agar mudah dibaca.

Serial adalah jembatan awal antara board dan manusia.

Tanpa serial, siswa sering menebak-nebak. Dengan serial, siswa bisa melihat proses internal program.

Kalimat penting:

Serial Monitor membuat program yang tidak terlihat menjadi terlihat.

5.2 Apa Itu Komunikasi Serial

Komunikasi serial adalah cara mengirim data secara berurutan melalui jalur komunikasi. Data dikirim sedikit demi sedikit, lalu diterima oleh perangkat lain.

Pada WeMOS mini D1, komunikasi serial umum terjadi melalui kabel USB. Komputer melihat board sebagai *serial device*. Arduino IDE kemudian menggunakan *Serial Monitor* untuk membaca dan mengirim pesan.

Dalam program Arduino, komunikasi serial biasanya dimulai dengan:

```
Serial.begin(9600);
```

`Serial.begin(baud)` digunakan untuk mengatur kecepatan komunikasi serial dalam satuan *baud*. Kecepatan ini harus sesuai dengan pengaturan di *Serial Monitor*. ([Arduino](#))

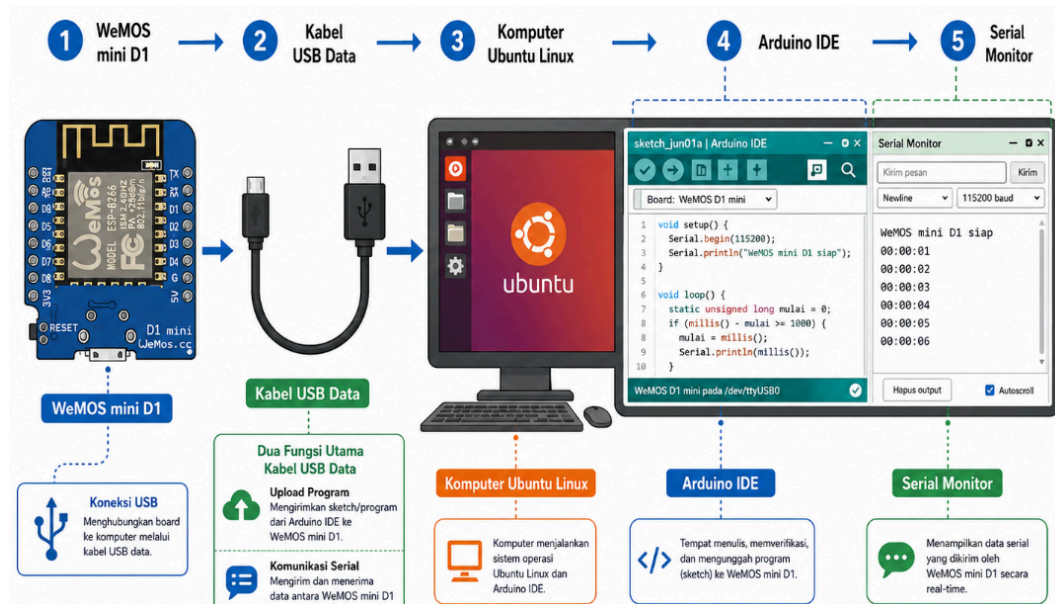
Contoh pengiriman pesan:

```
Serial.println("WeMOS mini D1 siap belajar IoT");
```

`Serial.print()` dan `Serial.println()` mengirim data ke *serial port* dalam bentuk teks yang mudah dibaca manusia. ([Arduino](#))

Pola sederhana komunikasi serial:

```
WeMOS mini D1 → Serial Port → Komputer  
Komputer → Serial Port → WeMOS mini D1
```



Gambar 5.1 Kabel USB data menghubungkan WeMOS mini D1 ke komputer untuk upload program dan komunikasi serial.

5.3 Fungsi TX dan RX pada WeMOS mini D1

Pada komunikasi serial, ada dua istilah penting:

- **TX** = *Transmit*, mengirim data.
- **RX** = *Receive*, menerima data.

Pada WeMOS mini D1:

- **TX** digunakan untuk mengirim data dari board.
- **RX** digunakan untuk menerima data ke board.

Saat board terhubung ke komputer melalui USB, chip USB-to-serial membantu menerjemahkan komunikasi antara WeMOS mini D1 dan komputer.

Aturan dasar TX/RX:

- TX perangkat pertama menuju RX perangkat kedua.
- RX perangkat pertama menerima dari TX perangkat kedua.
- GND harus sama jika memakai koneksi serial langsung antar perangkat.

Namun dalam praktik awal buku ini, siswa cukup memakai kabel USB. Tidak perlu menyambungkan pin TX/RX secara manual.

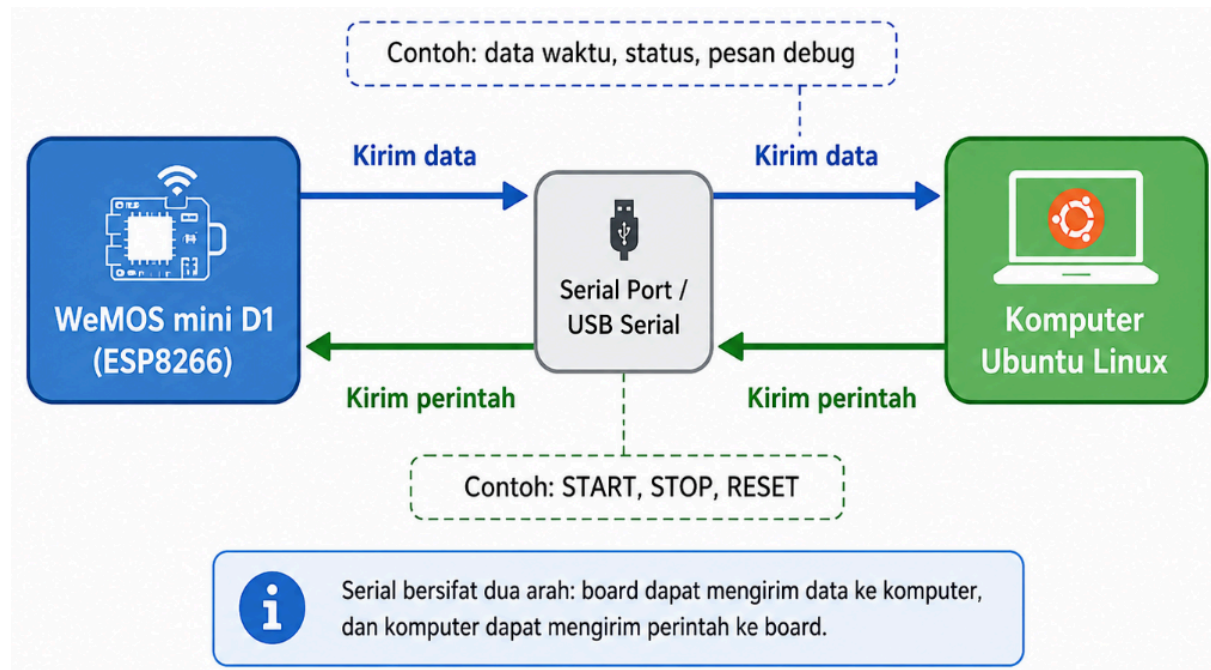
Catatan kritis:

Jangan memakai pin TX/RX untuk rangkaian lain saat sedang debugging serial, karena dapat mengganggu upload dan komunikasi.

5.4 Serial Monitor sebagai Alat Debugging

Serial Monitor adalah jendela untuk melihat pesan dari board dan mengirim perintah ke board.

Arduino IDE memiliki *Serial Monitor* untuk membaca data serial dari board, mengirim data, dan melihat keluaran program secara langsung. ([Dokumentasi Arduino](#))



Gambar 5.2 Komunikasi serial memungkinkan board mengirim data ke komputer dan menerima perintah dari komputer.

Serial Monitor berguna untuk:

- Melihat pesan program.
- Melihat nilai sensor.
- Melihat status tombol.
- Melihat nilai potensiometer.
- Melihat status Wi-Fi.
- Melihat status MQTT.
- Melihat pesan error.
- Menguji perintah sederhana.

Contoh pesan debugging:

```
Program mulai...  
Detik: 1  
Detik: 2  
Detik: 3
```

Pada bab ini, Serial Monitor dipakai untuk membuat jam digital sederhana.

Kesalahan umum:

- Baud rate di kode dan Serial Monitor tidak sama.
- Port serial salah.
- Board belum ter-upload program.
- Kabel USB hanya untuk *charging*.
- Serial Monitor belum dibuka.
- Program tidak memanggil `Serial.begin()`.

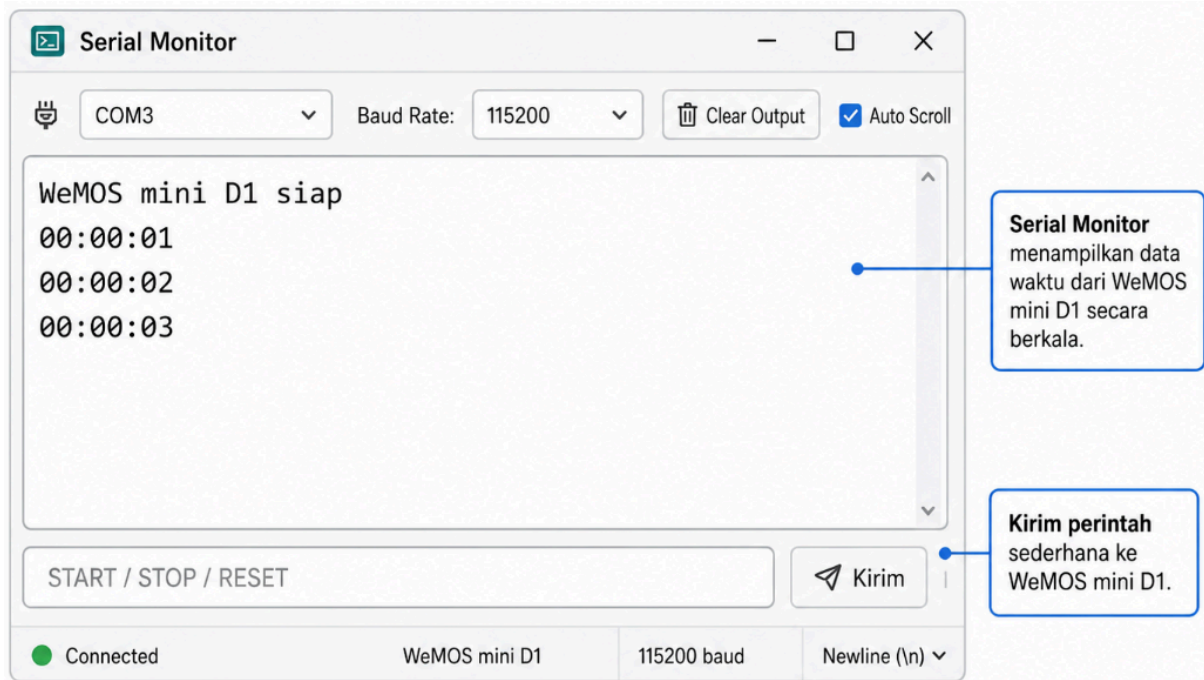
Prinsip debugging:

Jika tidak terlihat di Serial Monitor, tampilkan dulu dengan `Serial.println()`.

5.5 Format Data Waktu: Jam, Menit, Detik

Jam digital sederhana membutuhkan tiga bagian:

- **Jam**
- **Menit**
- **Detik**



Gambar 5.3 Serial Monitor menampilkan data waktu dari WeMOS mini D1 dan dapat mengirim perintah sederhana ke board.

Format umum:

HH:MM:SS

Contoh:

00:00:01
00:00:02
00:01:00
01:15:09

Agar mudah dibaca, angka harus dua digit.

Contoh:

- 1 detik ditulis 01, bukan 1.
- 5 menit ditulis 05, bukan 5.
- 9 jam ditulis 09, bukan 9.

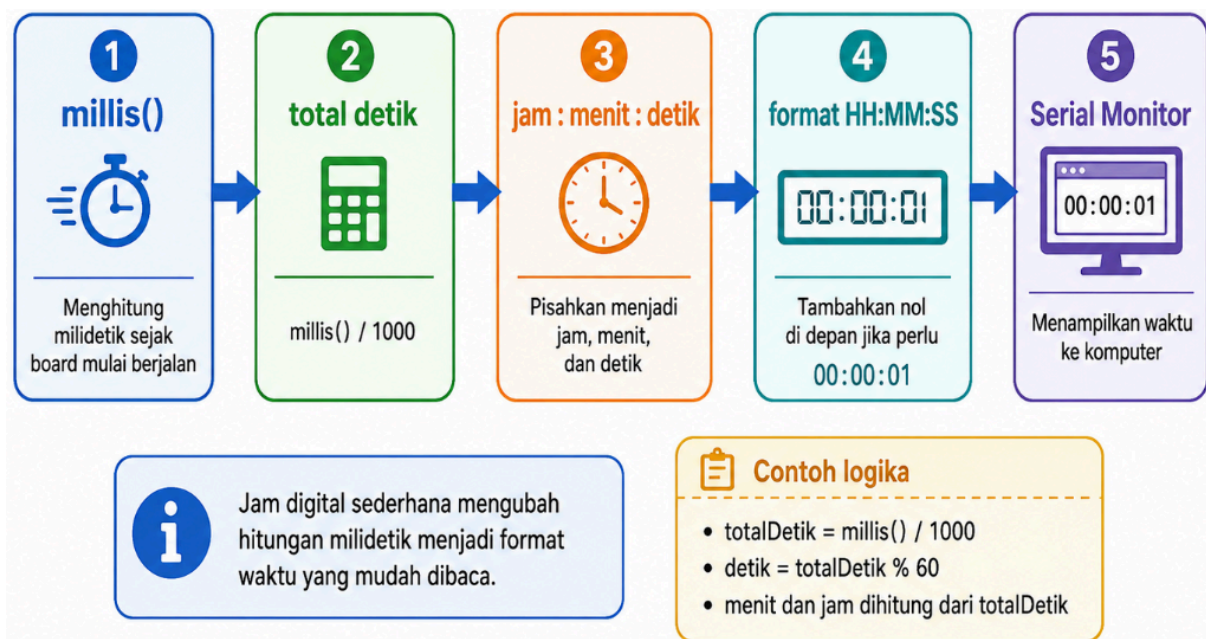
Dalam program, waktu dapat dihitung dari total detik.

Rumus sederhana:

```
totalDetik = waktuMillis / 1000
detik = totalDetik % 60
menit = (totalDetik / 60) % 60
jam = (totalDetik / 3600) % 24
```

Catatan:

Ini bukan jam real-time. Ini hanya jam hitung sejak program dimulai atau sejak tombol RESET diberikan.



Gambar 5.4 Jam digital sederhana mengubah hitungan milidetik menjadi format waktu yang mudah dibaca.

5.6 Membuat Jam Digital Sederhana dengan millis()

Fungsi `millis()` mengembalikan jumlah milidetik sejak board mulai menjalankan program. Dokumentasi Arduino mencatat bahwa nilai ini akan *overflow* atau kembali ke nol setelah periode tertentu, sehingga harus dipahami sebagai penghitung waktu relatif, bukan jam kalender. ([Dokumentasi Arduino](#))

Contoh dasar:

```
void setup() {  
    Serial.begin(9600);  
}  
  
void loop() {  
    unsigned long waktu = millis();  
  
    Serial.print("Millis: ");  
    Serial.println(waktu);  
  
    delay(1000);  
}
```

Kode di atas menampilkan jumlah milidetik sejak program mulai.

Untuk jam digital, nilai milidetik diubah menjadi detik, menit, dan jam.

Konsep penting:

- `millis()` bukan jam dinding.
- `millis()` menghitung waktu sejak board menyala.
- Jika board di-reset, hitungan kembali dari awal.
- Jika listrik mati, waktu hilang.
- Jika butuh waktu nyata, gunakan RTC atau NTP.

Kalimat penting:

`millis()` cocok untuk belajar konsep waktu, tetapi bukan pengganti jam real-time.

5.7 Mengirim Format HH:MM:SS ke Komputer

Kode berikut membuat jam digital sederhana dan mengirim format **HH:MM:SS** ke Serial Monitor.

```
unsigned long waktuAwal = 0;

void setup() {
  Serial.begin(9600);
  waktuAwal = millis();

  Serial.println("Jam digital sederhana dimulai");
}

void loop() {
  unsigned long waktuSekarang = millis();
  unsigned long totalDetik = (waktuSekarang - waktuAwal) /
1000;

  int detik = totalDetik % 60;
  int menit = (totalDetik / 60) % 60;
  int jam = (totalDetik / 3600) % 24;

  if (jam < 10) Serial.print("0");
  Serial.print(jam);
  Serial.print(":");

  if (menit < 10) Serial.print("0");
  Serial.print(menit);
  Serial.print(":");

  if (detik < 10) Serial.print("0");
  Serial.println(detik);

  delay(1000);
}
```

Hasil yang diharapkan:

```
Jam digital sederhana dimulai
00:00:00
00:00:01
00:00:02
00:00:03
```

Penjelasan ringkas:

- `waktuAwal` menyimpan waktu mulai.
- `millis()` membaca waktu berjalan.
- Selisih waktu diubah menjadi detik.
- Detik dihitung dengan `% 60`.
- Menit dihitung dari total detik.
- Jam dihitung dari total detik.
- Angka di bawah 10 diberi `0` di depan.

5.8 Membaca Perintah START, STOP, RESET dari Komputer

Komunikasi serial tidak hanya satu arah. Komputer juga dapat mengirim perintah ke WeMOS mini D1.

Perintah yang digunakan:

- **START** untuk menjalankan jam.
- **STOP** untuk menghentikan jam.
- **RESET** untuk mengulang waktu dari nol.

Fungsi `Serial.available()` dapat digunakan untuk mengecek apakah ada data serial yang masuk dari komputer. ([Arduino](#))

Kode contoh:

```
unsigned long waktuAwal = 0;
unsigned long waktuStop = 0;
bool berjalan = true;

void setup() {
  Serial.begin(9600);
  waktuAwal = millis();

  Serial.println("Ketik START, STOP, atau RESET");
}

void loop() {
  if (Serial.available() > 0) {
    String perintah = Serial.readStringUntil('\n');
    perintah.trim();

    if (perintah == "START") {
      berjalan = true;
      waktuAwal = millis() - waktuStop;
      Serial.println("Jam berjalan");
    }

    else if (perintah == "STOP") {
      berjalan = false;
      waktuStop = millis() - waktuAwal;
      Serial.println("Jam berhenti");
    }

    else if (perintah == "RESET") {
      waktuAwal = millis();
      waktuStop = 0;
      berjalan = true;
    }
  }
}
```

```

        Serial.println("Jam di-reset");
    }

    else {
        Serial.println("Perintah tidak dikenal");
    }
}

unsigned long totalDetik;

if (berjalan) {
    totalDetik = (millis() - waktuAwal) / 1000;
} else {
    totalDetik = waktuStop / 1000;
}

int detik = totalDetik % 60;
int menit = (totalDetik / 60) % 60;
int jam = (totalDetik / 3600) % 24;

if (jam < 10) Serial.print("0");
Serial.print(jam);
Serial.print(":");

if (menit < 10) Serial.print("0");
Serial.print(menit);
Serial.print(":");

if (detik < 10) Serial.print("0");
Serial.println(detik);

delay(1000);
}

```

Hasil yang diharapkan:

Ketik START, STOP, atau RESET
00:00:01
00:00:02
STOP
Jam berhenti
00:00:02
START
Jam berjalan
00:00:03
RESET
Jam di-reset
00:00:00

Nilai penting:

Siswa belajar bahwa komputer dapat mengirim **perintah teks**, dan board dapat merespons secara logis.

5.9 Kelemahan Jam Berbasis millis() dan Pengembangan RTC/NTP

Jam berbasis `millis()` cocok untuk belajar, tetapi memiliki batas.

Kelemahan:

- Mulai dari nol setiap board menyala.
- Tidak mengetahui tanggal.
- Tidak mengetahui waktu nyata.
- Bisa bergeser karena akurasi clock board.
- Hilang saat board reset.
- Nilai `millis()` dapat *overflow* setelah waktu panjang. ([Dokumentasi Arduino](#))

Untuk proyek lebih serius, ada dua pengembangan:

1. RTC

RTC atau *Real-Time Clock* adalah modul jam fisik. Modul ini biasanya memiliki baterai kecil agar waktu tetap berjalan walau board mati.

Cocok untuk:

- Jam lokal.
- Data logger.
- Sistem yang tidak selalu terhubung internet.

2. NTP

NTP atau *Network Time Protocol* mengambil waktu dari server waktu melalui jaringan. Pada ESP8266, konsep NTP umum digunakan untuk mendapatkan waktu dari server setelah perangkat tersambung ke jaringan. (tttapa.github.io)

Cocok untuk:

- Jam IoT berbasis Wi-Fi.
- Dashboard waktu.
- Timestamp data sensor.
- Sistem yang terhubung jaringan.

Prinsip pengembangan:

Gunakan `millis()` untuk belajar waktu relatif. Gunakan RTC atau NTP untuk waktu nyata.

5.10 Latihan, Tugas, dan Checklist Hasil

Latihan Bab 5

1. Apa fungsi komunikasi serial?
2. Apa perbedaan TX dan RX?
3. Mengapa Serial Monitor penting untuk debugging?
4. Apa fungsi `Serial.begin(9600)`?
5. Apa fungsi `Serial.println()`?
6. Apa yang dikembalikan oleh `millis()`?
7. Mengapa format `HH:MM:SS` perlu dua digit?
8. Apa fungsi perintah `START`?
9. Apa fungsi perintah `STOP`?
10. Apa fungsi perintah `RESET`?
11. Apa kelemahan jam berbasis `millis()`?
12. Kapan sebaiknya memakai RTC?
13. Kapan sebaiknya memakai NTP?

Tugas Praktik Bab 5

Buat tiga program:

1. **Menampilkan angka detik di Serial Monitor.**
2. **Membuat jam digital sederhana format `HH:MM:SS`.**
3. **Membuat jam digital dengan perintah `START`, `STOP`, `RESET`.**

Laporan singkat berisi:

- Screenshot kode.
- Screenshot Serial Monitor.
- Port serial yang dipakai.
- Baud rate yang dipakai.
- Contoh keluaran `HH:MM:SS`.
- Contoh perintah `START`, `STOP`, `RESET`.
- Masalah yang muncul.
- Solusi yang dilakukan.

Checklist Hasil

Siswa dinyatakan berhasil jika mampu:

- Menjelaskan fungsi komunikasi serial.
- Menjelaskan TX dan RX.
- Mengaktifkan Serial Monitor.
- Menampilkan teks dari WeMOS mini D1 ke komputer.
- Membuat penghitung detik.
- Membuat format `HH:MM:SS`.
- Mengirim perintah dari komputer ke board.
- Menjelaskan kelemahan `millis()`.
- Menjelaskan ide pengembangan ke RTC atau NTP.

Catatan Akhir Bab 5

Siswa yang memahami serial akan lebih mudah memahami MQTT, karena keduanya sama-sama tentang mengirim pesan; bedanya, serial berbicara lewat kabel USB, sedangkan MQTT berbicara lewat jaringan.

BAB 6 — Menghubungkan WeMOS mini D1 ke Wi-Fi dan Jaringan Lokal

6.1 Mengapa Wi-Fi Penting dalam IoT

Pada bab sebelumnya, WeMOS mini D1 sudah bisa membaca input, mengendalikan LED, membaca potensiometer, dan mengirim data ke komputer melalui *serial port*. Namun semua itu masih bersifat lokal.

Agar menjadi IoT, board harus bisa masuk ke jaringan.

Wi-Fi membuat WeMOS mini D1 dapat:

- Mengirim data ke komputer atau server.
- Menerima perintah dari perangkat lain.
- Menjadi bagian dari jaringan sekolah.
- Mengakses MQTT Broker di Ubuntu Server.
- Mengirim nilai sensor ke dashboard Android.
- Menjalankan web server sederhana untuk pengantar komunikasi lokal.

Perubahan penting Bab 6:

WeMOS mini D1 berubah dari alat lokal menjadi node jaringan.

Dalam sistem buku ini, arah akhirnya adalah:

Sensor → WeMOS mini D1 → Wi-Fi → MQTT Broker di Ubuntu Server →
Android Dashboard

Bab 6 fokus pada bagian:

WeMOS mini D1 → Wi-Fi → Jaringan Lokal



Gambar 6.1 WeMOS mini D1 tersambung ke router sebagai station dan dapat diuji dari laptop Ubuntu dalam jaringan yang sama.

6.2 Konsep Wi-Fi pada ESP8266

WeMOS mini D1 memakai chip ESP8266EX. Board ini sudah memiliki Wi-Fi, sehingga tidak membutuhkan modul Wi-Fi tambahan.

ESP8266 Arduino Core menyediakan library **ESP8266WiFi** untuk koneksi Wi-Fi, pemindaian jaringan, status koneksi, TCP/UDP, HTTP server, dan fitur jaringan lain. Dokumentasi ESP8266 Arduino Core menjelaskan bahwa library ini menyediakan fungsi Wi-Fi seperti `WiFi.status()`, koneksi ke jaringan, pemindaian, dan komunikasi jaringan. ([ESP8266 Arduino Core](#))

Dalam praktik dasar, ada tiga konsep penting:

- **SSID**
Nama jaringan Wi-Fi.
- **Password**
Kata sandi jaringan Wi-Fi.
- **IP address**
Alamat WeMOS mini D1 setelah berhasil tersambung ke router.

Contoh:

```
SSID      : Lab-IoT-SMK
Password  : *****
IP board  : 192.168.1.45
```

Jika board belum memiliki IP address, perangkat lain belum bisa menghubunginya di jaringan lokal.

Kalimat penting:

Dalam jaringan, alamat IP adalah identitas perangkat.

6.3 Mode Station: WeMOS mini D1 Tersambung ke Router

Mode yang dipakai dalam bab ini adalah **station mode**.

Dalam *station mode*, WeMOS mini D1 bertindak seperti HP atau laptop: ia tersambung ke router Wi-Fi yang sudah ada.

Dokumentasi ESP8266 Arduino Core menjelaskan bahwa koneksi mode station dilakukan dengan fungsi:

```
WiFi.begin(ssid, password);
```

Fungsi ini memakai SSID dan password agar modul dapat tersambung ke *access point*. ([ESP8266 Arduino Core](#))

Arsitektur sederhana:

```
WeMOS mini D1 → Router Wi-Fi → Laptop Ubuntu / Ubuntu Server  
/ Android
```

Mode ini cocok untuk lab SMK karena:

- Router sudah biasa tersedia.
- Banyak perangkat bisa berada dalam jaringan yang sama.
- WeMOS mini D1 bisa mengirim data ke Ubuntu Server.
- Android bisa membaca data dari server.
- Guru dapat menguji koneksi dari laptop.

Catatan kritis:

Pastikan WeMOS mini D1 dan komputer berada pada **jaringan yang sama**, agar uji koneksi lokal lebih mudah.

6.4 Praktik 1 — Wi-Fi Scanner

Sebelum tersambung ke Wi-Fi, siswa perlu belajar memindai jaringan yang tersedia.

Dokumentasi ESP8266 Arduino Core menyediakan contoh **WiFiScan** di Arduino IDE melalui menu **Examples** → **ESP8266WiFi** → **WiFiScan**. Contoh ini memperlihatkan cara ESP8266 menemukan jaringan Wi-Fi di sekitarnya. ([ESP8266 Arduino Core](#))

Tujuan

Siswa melihat daftar SSID yang terdeteksi oleh WeMOS mini D1.

Kode Wi-Fi Scanner

```
#include <ESP8266WiFi.h>

void setup() {
  Serial.begin(9600);
  delay(1000);

  Serial.println();
  Serial.println("Memindai jaringan Wi-Fi...");
}

void loop() {
  int jumlahJaringan = WiFi.scanNetworks();

  if (jumlahJaringan == 0) {
    Serial.println("Tidak ada jaringan ditemukan");
  } else {
    Serial.print(jumlahJaringan);
    Serial.println(" jaringan ditemukan:");

    for (int i = 0; i < jumlahJaringan; i++) {
      Serial.print(i + 1);
      Serial.print(". ");
      Serial.print(WiFi.SSID(i));
      Serial.print(" | Sinyal: ");
      Serial.print(WiFi.RSSI(i));
      Serial.println(" dBm");
    }
  }

  Serial.println();
  delay(5000);
}
```

Hasil yang diharapkan

Serial Monitor menampilkan:

```
Memindai jaringan Wi-Fi...  
3 jaringan ditemukan:  
1. Lab-IoT-SMK | Sinyal: -45 dBm  
2. Ruang-Guru | Sinyal: -70 dBm  
3. Kantin | Sinyal: -82 dBm
```

Makna RSSI secara sederhana:

- Angka mendekati 0 biasanya lebih kuat.
- **-45 dBm** lebih kuat daripada **-80 dBm**.
- Sinyal lemah dapat membuat koneksi tidak stabil.

6.5 Praktik 2 — Koneksi ke SSID dan Password

Setelah tahu jaringan tersedia, siswa mencoba menyambungkan WeMOS mini D1 ke Wi-Fi lab.

Tujuan

Siswa dapat menghubungkan WeMOS mini D1 ke router dan menampilkan status koneksi.

Kode koneksi Wi-Fi

```
#include <ESP8266WiFi.h>

const char* ssid = "Lab-IoT-SMK";
const char* password = "password_wifi_lab";

void setup() {
    Serial.begin(9600);
    delay(1000);

    Serial.println();
    Serial.print("Menghubungkan ke ");
    Serial.println(ssid);

    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }

    Serial.println();
    Serial.println("Wi-Fi tersambung");
    Serial.print("Alamat IP: ");
    Serial.println(WiFi.localIP());
}

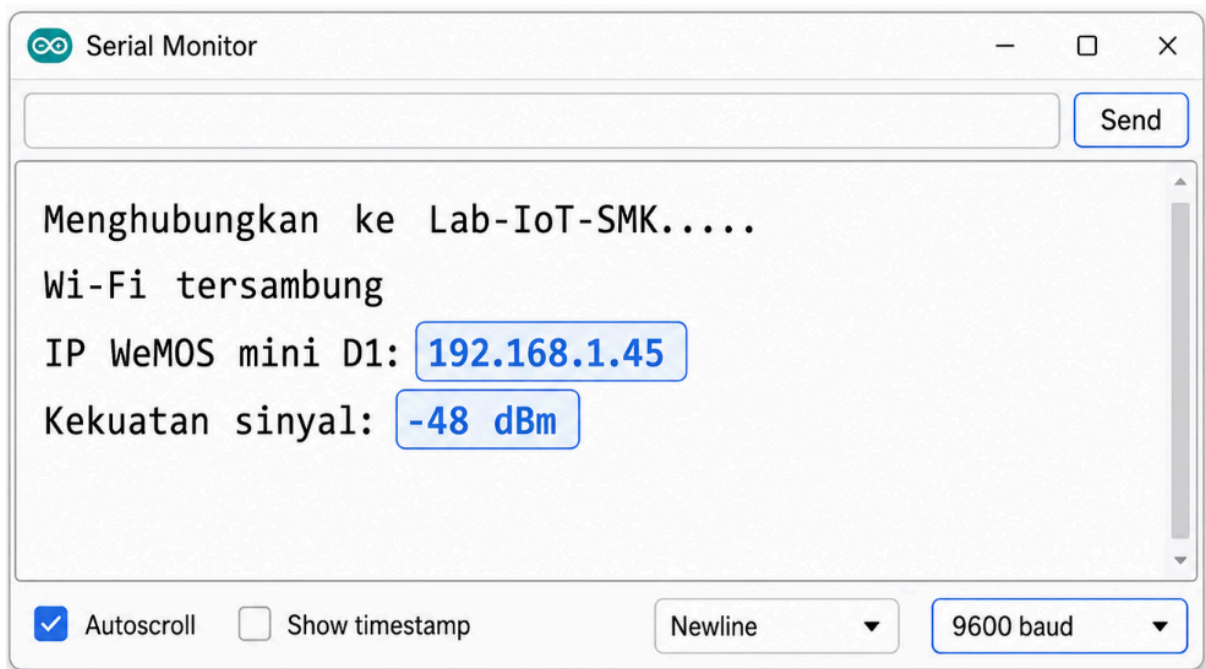
void loop() {
}
```

Penjelasan singkat

- `#include <ESP8266WiFi.h>`
Memanggil library Wi-Fi untuk ESP8266.
- `ssid`
Menyimpan nama jaringan Wi-Fi.
- `password`
Menyimpan kata sandi Wi-Fi.
- `WiFi.mode(WIFI_STA)`
Mengatur board sebagai *station*.
- `WiFi.begin(ssid, password)`
Memulai koneksi ke router.
- `WiFi.status()`
Mengecek status koneksi.
- `WiFi.localIP()`
Menampilkan alamat IP board.

Aturan keamanan:

Jangan menulis password Wi-Fi sekolah di laporan publik. Samarkan jika perlu.



Gambar 6.2 Serial Monitor menampilkan status koneksi Wi-Fi, alamat IP, dan kekuatan sinyal WeMOS mini D1.

6.6 Membaca Alamat IP dan Status Koneksi

Alamat IP adalah informasi terpenting setelah board berhasil tersambung ke Wi-Fi.

Contoh tampilan:

```
Wi-Fi tersambung  
Alamat IP: 192.168.1.45
```

Dokumentasi ESP8266 Arduino Core menjelaskan bahwa `WiFi.status()` dapat digunakan untuk membaca status koneksi. Beberapa status penting antara lain `WL_CONNECTED` saat koneksi berhasil, `WL_NO_SSID_AVAIL` jika SSID tidak ditemukan, `WL_CONNECT_FAILED` jika koneksi gagal, dan `WL_CONNECT_WRONG_PASSWORD` jika password salah. ([ESP8266 Arduino Core](#))

Kode status koneksi lebih jelas

```
#include <ESP8266WiFi.h>

const char* ssid = "Lab-IoT-SMK";
const char* password = "password_wifi_lab";

void setup() {
  Serial.begin(9600);
  delay(1000);

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  Serial.print("Mencoba koneksi");

  int hitung = 0;
  while (WiFi.status() != WL_CONNECTED && hitung < 30) {
    delay(500);
    Serial.print(".");
    hitung++;
  }

  Serial.println();

  if (WiFi.status() == WL_CONNECTED) {
    Serial.println("Status: TERHUBUNG");
    Serial.print("IP WeMOS mini D1: ");
    Serial.println(WiFi.localIP());
    Serial.print("Kekuatan sinyal: ");
    Serial.print(WiFi.RSSI());
    Serial.println(" dBm");
  } else {
```

```
        Serial.print("Status gagal. Kode: ");  
        Serial.println(WiFi.status());  
    }  
}  
  
void loop() {  
}
```

Kalimat penting:

Selalu tampilkan IP di Serial Monitor, karena IP adalah pintu untuk menguji komunikasi lokal.

6.7 Tes Koneksi Jaringan Lokal

Setelah IP muncul, komputer Ubuntu Linux dapat menguji koneksi ke WeMOS mini D1.

Contoh IP board:

```
192.168.1.45
```

Tes dari komputer:

```
ping 192.168.1.45
```

Jika berhasil, komputer menerima balasan.

Contoh:

```
64 bytes from 192.168.1.45: icmp_seq=1 ttl=255 time=5.2 ms
64 bytes from 192.168.1.45: icmp_seq=2 ttl=255 time=4.8 ms
```

Jika gagal, periksa:

- IP board benar atau tidak.
- Komputer dan WeMOS ada di jaringan yang sama atau tidak.
- Router memblokir komunikasi antar perangkat atau tidak.
- Board masih menyala atau tidak.
- Wi-Fi tersambung atau putus.

Catatan kritis:

Beberapa router mengaktifkan fitur isolasi klien. Jika fitur ini aktif, sesama perangkat Wi-Fi tidak bisa saling berkomunikasi. Ini sering membuat siswa mengira kodenya salah, padahal masalahnya ada di konfigurasi router.

6.8 Praktik 3 — Halaman Web Sederhana dari WeMOS mini D1

Sebelum masuk MQTT, siswa dapat mencoba halaman web sederhana. Ini membuat siswa paham bahwa WeMOS mini D1 bisa menerima permintaan dari browser di jaringan lokal.

ESP8266 Arduino Core menyediakan contoh server. Dokumentasinya menjelaskan bahwa ESP8266 dapat menjalankan web server sederhana dan merespons permintaan dari komputer dengan browser. ([ESP8266 Arduino Core](#))

Tujuan

Siswa membuka alamat IP WeMOS mini D1 dari browser dan melihat pesan sederhana.

Kode web server sederhana

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>

const char* ssid = "Lab-IoT-SMK";
const char* password = "password_wifi_lab";

ESP8266WebServer server(80);

void halamanUtama() {
    server.send(200, "text/html",
        "<h1>WeMOS mini D1 Aktif</h1>"
        "<p>Halo dari jaringan lokal SMK.</p>"
    );
}

void setup() {
    Serial.begin(9600);
    delay(1000);

    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, password);

    Serial.print("Menghubungkan ke Wi-Fi");
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }

    Serial.println();
    Serial.print("Buka alamat: http://");
    Serial.println(WiFi.localIP());
}
```

```
server.on("/", halamanUtama);  
server.begin();  
  
Serial.println("Web server berjalan");  
}  
  
void loop() {  
    server.handleClient();  
}
```

Hasil yang diharapkan

Serial Monitor menampilkan:

```
Buka alamat: http://192.168.1.45  
Web server berjalan
```

Browser menampilkan:

```
WeMOS mini D1 Aktif  
Halo dari jaringan lokal SMK.
```

Catatan:

Web server ini hanya pengantar. Untuk sistem IoT buku ini, komunikasi utama tetap diarahkan ke **MQTT** pada bab berikutnya.

6.9 Kesalahan Umum Wi-Fi dan Troubleshooting

1. Salah password

Gejala:

- Koneksi tidak berhasil.
- Status bisa menunjukkan password salah.
- Serial Monitor hanya menampilkan titik terus.

Solusi:

- Ketik ulang password.
- Perhatikan huruf besar/kecil.
- Hindari spasi tidak sengaja.
- Gunakan jaringan lab khusus.

2. SSID tidak ditemukan

Gejala:

- Wi-Fi scanner tidak menemukan jaringan.
- Status menunjukkan SSID tidak tersedia.

Solusi:

- Pastikan router menyala.
- Dekatkan board ke router.
- Pastikan jaringan 2.4 GHz tersedia.
- Ulangi Wi-Fi scanner.

3. Sinyal lemah

Gejala:

- Koneksi sering putus.
- IP kadang muncul kadang tidak.
- Halaman web lambat.

Solusi:

- Dekatkan board ke router.
- Kurangi penghalang fisik.
- Pakai catu daya stabil.
- Hindari router terlalu jauh dari meja praktek.

4. IP berubah

Gejala:

- Kemarin IP bisa dibuka, hari ini tidak.
- Android atau komputer tidak menemukan board.

Solusi:

- Selalu cek `WiFi.localIP()` di Serial Monitor.
- Catat IP terbaru.
- Untuk proyek lanjutan, gunakan DHCP reservation di router atau IP statis.

5. Jaringan tidak stabil

Gejala:

- Ping putus-putus.
- Web server kadang terbuka, kadang gagal.
- Serial Monitor menunjukkan reconnect.

Solusi:

- Pastikan daya USB stabil.
- Hindari power dari port lemah.
- Kurangi beban pada board.
- Gunakan router khusus lab jika memungkinkan.

Prinsip troubleshooting:

Jika Wi-Fi gagal, cek urutan ini: SSID → password → sinyal → IP → router → daya.

6.10 Batasan ESP8266 dan Persiapan Menuju MQTT

WeMOS mini D1 berbasis ESP8266 sangat cocok untuk belajar IoT sederhana. Namun siswa harus tahu batasannya.

Batasan ESP8266:

- Memori terbatas.
- Pin lebih sedikit.
- Input analog hanya satu.
- Perlu hati-hati dengan pin saat *boot*.
- Tidak cocok menjalankan aplikasi server yang berat.
- Lebih cocok sebagai node sensor dan aktuator.

ESP8266 Arduino Core mendukung banyak fitur jaringan seperti Wi-Fi, TCP/UDP, HTTP, mDNS, DNS server, OTA, file system flash, dan berbagai pustaka lain. Namun untuk buku SMK, fitur perlu dipilih yang paling relevan dan tidak membuat siswa tenggelam dalam kompleksitas. ([GitHub](#))

Arah desain buku:

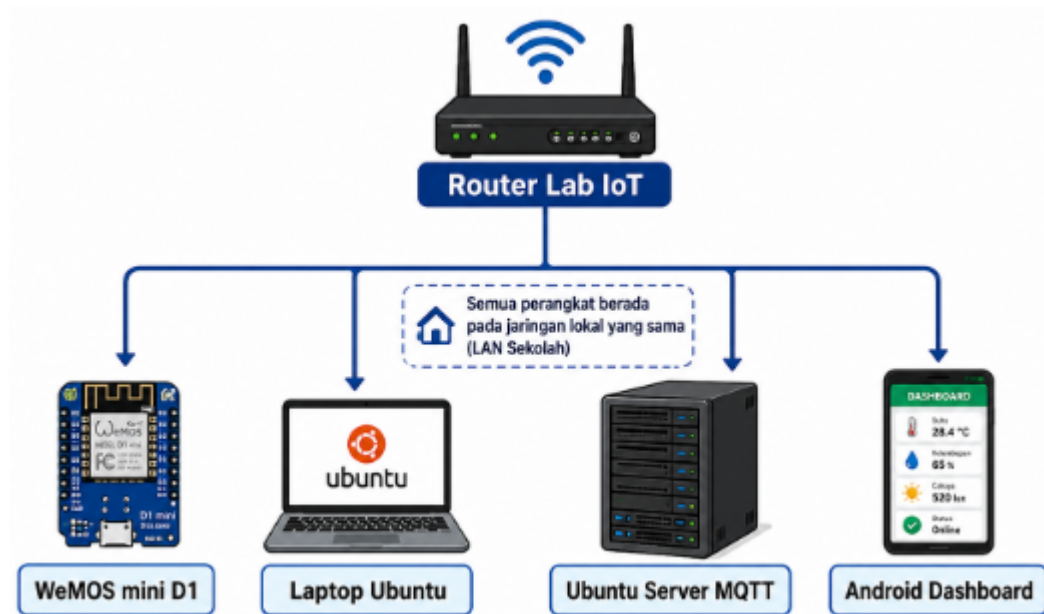
- WeMOS mini D1 membaca sensor.
- WeMOS mini D1 mengirim data melalui Wi-Fi.
- Ubuntu Server menjalankan Mosquitto MQTT Broker.
- Android membaca dan mengontrol melalui dashboard.

Bab berikutnya akan masuk ke MQTT. Jika Bab 6 berhasil, siswa sudah punya syarat utama:

- Board tersambung Wi-Fi.
- Board memiliki IP address.
- Komputer bisa menguji koneksi.
- Siswa paham bahwa data bisa bergerak di jaringan lokal.

Penutup Bab 6:

Ketika WeMOS mini D1 sudah tersambung Wi-Fi dan memiliki alamat IP, ia tidak lagi hanya board kecil di meja praktek; ia sudah menjadi bagian dari jaringan IoT.



Gambar 6.3 Jaringan lokal sekolah menjadi tempat WeMOS mini D1, laptop, server, dan Android saling bertukar data.

Latihan Bab 6

1. Apa fungsi Wi-Fi pada WeMOS mini D1?
2. Apa yang dimaksud dengan *station mode*?
3. Apa fungsi SSID?
4. Apa fungsi password Wi-Fi?
5. Mengapa IP address penting?
6. Apa fungsi `WiFi.begin(ssid, password)`?
7. Apa fungsi `WiFi.localIP()`?
8. Mengapa Serial Monitor harus menampilkan status koneksi?
9. Apa penyebab umum Wi-Fi gagal tersambung?
10. Mengapa jaringan lokal sekolah cocok untuk belajar IoT?
11. Mengapa ESP8266 tidak cocok menjadi server berat?
12. Apa hubungan Bab 6 dengan Bab MQTT berikutnya?

Tugas Praktik Bab 6

Buat tiga program:

1. **Wi-Fi scanner**
 - Menampilkan daftar SSID.
 - Menampilkan kekuatan sinyal.
2. **Koneksi Wi-Fi lab**
 - Menyambungkan WeMOS mini D1 ke router.
 - Menampilkan IP address.
 - Menampilkan RSSI.
3. **Web server sederhana**
 - Menampilkan halaman:
WeMOS mini D1 Aktif
 - Halo dari jaringan lokal SMK

Laporan singkat berisi:

- Screenshot Serial Monitor.
- SSID yang dipakai.
- IP WeMOS mini D1.
- Hasil **ping** dari laptop Ubuntu.
- Screenshot halaman web sederhana.
- Masalah koneksi.
- Solusi yang dilakukan.

Target akhir Bab 6:

Siswa mampu menghubungkan WeMOS mini D1 ke Wi-Fi, membaca IP address, menguji koneksi lokal, dan memahami bahwa board sudah menjadi node jaringan sebelum masuk ke MQTT.

BAB 7 — Ubuntu Server Lokal dan Mosquitto MQTT Broker

7.1 Mengapa IoT Membutuhkan Server Lokal

Pada bab sebelumnya, WeMOS mini D1 sudah bisa tersambung ke Wi-Fi dan memiliki alamat IP. Namun jaringan saja belum cukup. IoT membutuhkan tempat untuk **menerima**, **mengatur**, dan **meneruskan pesan**.

Di buku ini, pusat pesan itu adalah **Mosquitto MQTT Broker** yang berjalan di **Ubuntu Server lokal**.

Tanpa broker, setiap perangkat harus saling terhubung langsung. Ini cepat menjadi rumit. Dengan broker, sistem lebih rapi:

```
WeMOS mini D1 → MQTT Broker → Android Dashboard
Android Dashboard → MQTT Broker → WeMOS mini D1
Laptop Ubuntu → MQTT Broker → WeMOS mini D1
```

Server lokal membuat IoT lebih mudah dipahami siswa karena semua perangkat berada di jaringan sekolah. Tidak perlu bergantung pada layanan luar. Guru dan siswa dapat melihat langsung alamat IP, port, service, dan pesan MQTT.

Dokumen outline juga menekankan bahwa server lokal cocok untuk SMK karena murah, terbuka, tidak tergantung layanan luar, dan mendidik siswa tentang jaringan, IP address, SSH, service, serta firewall.

Kalimat penting:

Server lokal mengubah proyek IoT dari rangkaian sederhana menjadi sistem jaringan yang bisa dipantau dan dikontrol.

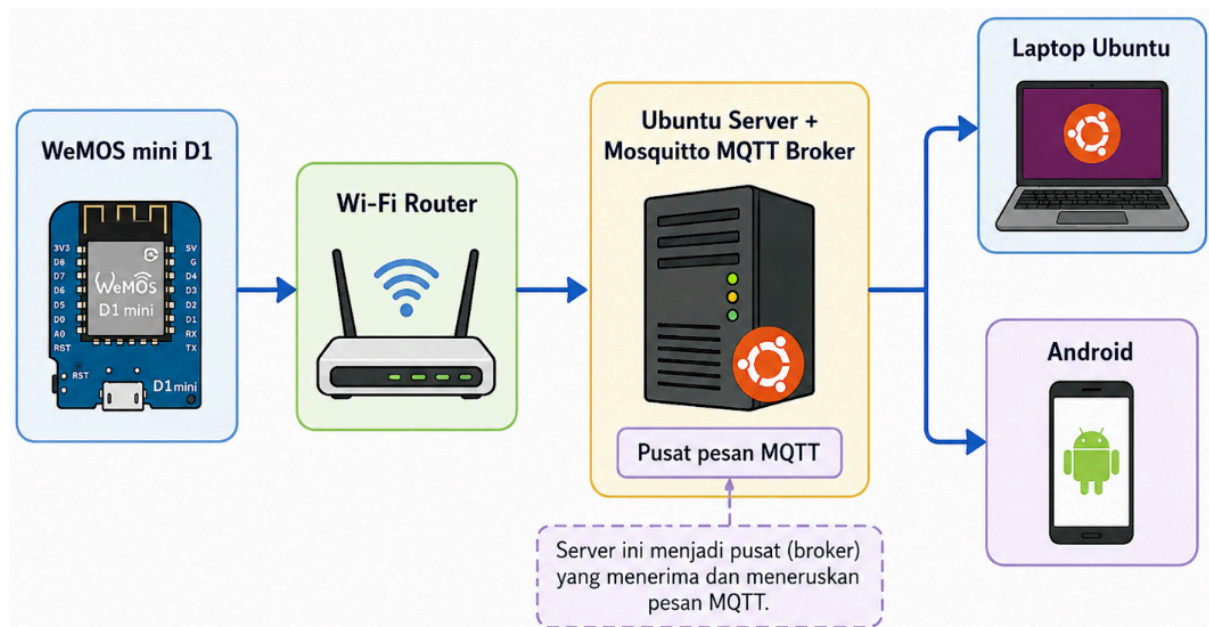
7.2 Peran Ubuntu Server dalam Sistem IoT SMK

Ubuntu Server berperan sebagai pusat layanan IoT di lab sekolah. Server ini dapat berjalan di PC bekas, mini PC, atau mesin kecil yang selalu menyala di jaringan lokal.

Peran utama Ubuntu Server:

- Menjalankan **Mosquitto MQTT Broker**.
- Menyediakan alamat IP tetap untuk perangkat lain.
- Menjadi pusat komunikasi antara WeMOS mini D1, laptop, dan Android.
- Menjadi tempat belajar SSH, service, port, firewall, dan troubleshooting.
- Menjadi pondasi untuk pengembangan berikutnya seperti Node-RED, InfluxDB, dan Grafana.

Dalam arsitektur buku:



Gambar 7.1 Mosquitto di Ubuntu Server menjadi pusat pesan antara WeMOS mini D1, laptop, dan Android dalam jaringan lokal SMK.

Prinsip desain:

WeMOS mini D1 membaca dan mengirim data. Ubuntu Server mengatur komunikasi. Android menampilkan dan mengirim perintah.

7.3 IP Statis, SSH, dan Struktur Server Lab

Server sebaiknya memakai **IP statis** agar alamatnya tidak berubah. Jika IP server berubah, WeMOS mini D1 dan Android tidak tahu harus terhubung ke mana.

Contoh:

```
IP Ubuntu Server MQTT : 192.168.1.100
Port MQTT              : 1883
Username MQTT          : smk_iot
```

Pada Ubuntu Server modern, konfigurasi jaringan biasanya memakai **Netplan**, yaitu sistem konfigurasi jaringan berbasis file YAML. Untuk server, alamat IP tetap penting agar layanan seperti SSH dan MQTT dapat diakses secara konsisten. ([DEV Community](#))

SSH dipakai untuk mengakses server dari laptop tanpa harus selalu berada di depan server.

Contoh akses:

```
ssh siswa@192.168.1.100
```

Penjelasan singkat:

- **ssh** adalah perintah untuk masuk ke server dari jarak jauh.
- **siswa** adalah nama pengguna di server.
- **192.168.1.100** adalah IP server.

Struktur folder lab yang disarankan:

```
/opt/iot-smk/
├── mosquitto/
├── kode-wemos/
├── catatan-topic/
└── log-praktik/
```

Catatan keamanan:

Gunakan user biasa. Jangan biasakan semua pekerjaan dilakukan sebagai root.

7.4 Instalasi Mosquitto MQTT Broker

Mosquitto adalah broker MQTT open source. Situs resmi Eclipse Mosquitto menjelaskan Mosquitto sebagai broker pesan open source yang mengimplementasikan MQTT versi 5.0, 3.1.1, dan 3.1, serta ringan untuk perangkat kecil sampai server penuh. ([Eclipse Mosquitto](#))

Instalasi dasar di Ubuntu Server:

```
sudo apt update
sudo apt install mosquitto mosquitto-clients
```

Aktifkan service:

```
sudo systemctl enable mosquitto
sudo systemctl start mosquitto
```

Cek status:

```
systemctl status mosquitto
```

Cek port MQTT:

```
ss -tulpn | grep 1883
```

Penjelasan singkat:

- **mosquitto** adalah broker.
- **mosquitto-clients** berisi alat uji seperti **mosquitto_pub** dan **mosquitto_sub**.
- **systemctl** dipakai untuk mengatur service.
- Port default MQTT adalah **1883**.

Kalimat penting:

Jika service Mosquitto belum berjalan, semua klien MQTT akan gagal terhubung.

7.5 Apa Itu MQTT

MQTT adalah protokol komunikasi ringan untuk IoT. MQTT bekerja dengan pola **publish/subscribe**. Perangkat tidak harus saling tahu alamat satu sama lain. Perangkat cukup berbicara melalui broker.

MQTT 5.0 adalah standar OASIS untuk protokol *publish/subscribe messaging* yang ringan, terutama cocok untuk IoT dan komunikasi mesin-ke-mesin. (docs.oasis-open.org)

Pola sederhana:

Publisher → Broker → Subscriber

Contoh:

```
WeMOS mini D1 publish nilai potensiometer
      ↓
Mosquitto menerima pesan
      ↓
Android subscribe dan menampilkan nilai
```

MQTT cocok untuk SMK karena:

- Konsepnya sederhana.
- Ringan untuk WeMOS mini D1.
- Cocok untuk data kecil.
- Mudah diuji dari laptop.
- Cocok untuk jaringan lokal.
- Banyak dipakai dalam sistem IoT nyata.

Kalimat kunci:

MQTT adalah bahasa percakapan perangkat IoT.

7.6 Broker, Publisher, Subscriber, Topic, Payload

Ada lima istilah utama yang harus dikuasai siswa.

1. Broker

Broker adalah pusat pesan. Dalam buku ini, broker adalah **Mosquitto** di Ubuntu Server.

Tugas broker:

- Menerima pesan.
- Menyimpan sementara sesuai aturan.
- Meneruskan pesan ke subscriber.
- Mengatur klien yang terhubung.

2. Publisher

Publisher adalah perangkat yang mengirim pesan.

Contoh:

```
WeMOS mini D1 mengirim nilai A0 ke topic  
smk/lab1/potensio
```

3. Subscriber

Subscriber adalah perangkat yang membaca pesan dari topic.

Contoh:

```
Laptop Ubuntu membaca topic smk/lab1/potensio  
Android membaca topic smk/lab1/led/status
```

4. Topic

Topic adalah alamat pesan.

Contoh:

```
smk/lab1/button  
smk/lab1/potensio  
smk/lab1/jam
```

5. Payload

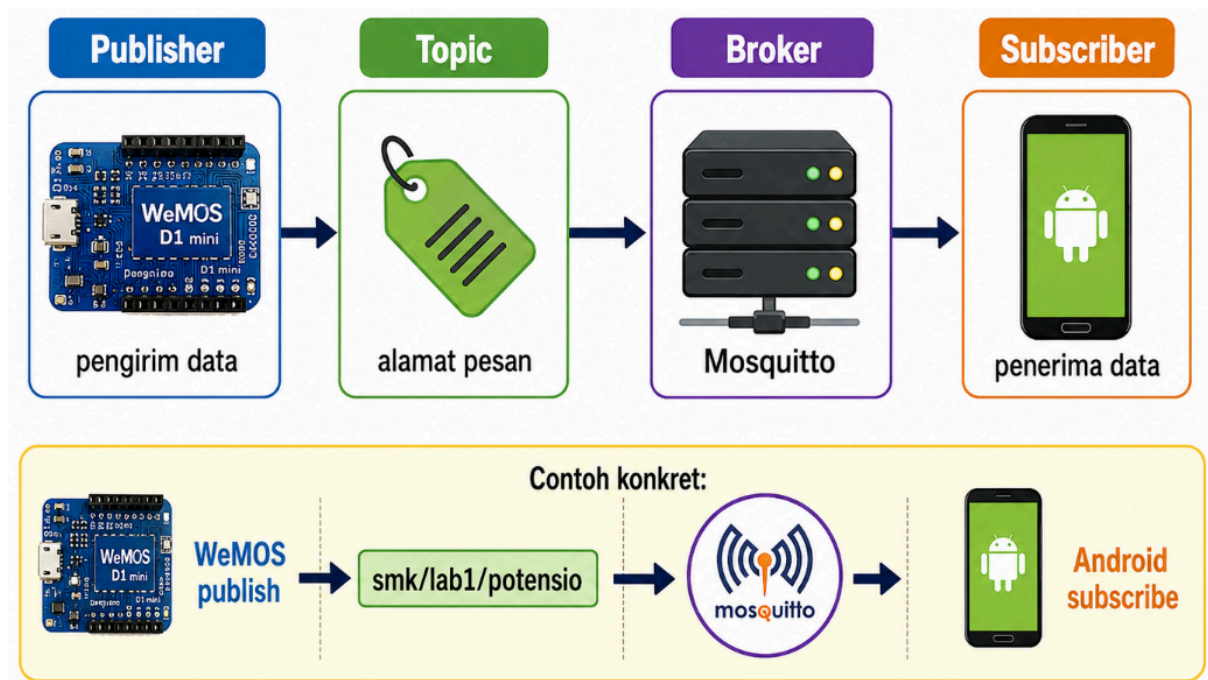
Payload adalah isi pesan.

Contoh:

```
Topic    : smk/lab1/led/status  
Payload  : ON
```

Prinsip:

Topic adalah alamat. **Payload** adalah isi.



Gambar 7.2 MQTT memisahkan pengirim dan penerima melalui broker sehingga perangkat IoT lebih mudah diatur.

7.7 Praktik 1 — Testing mosquitto_sub dan mosquitto_pub

Sebelum WeMOS mini D1 dipakai, broker harus diuji dari laptop atau server.

Buka terminal pertama sebagai subscriber:

```
mosquitto_sub -h localhost -t smk/lab1/test
```

Buka terminal kedua sebagai publisher:

```
mosquitto_pub -h localhost -t smk/lab1/test -m "Halo MQTT dari SMK"
```

Jika berhasil, terminal pertama menampilkan:

```
Halo MQTT dari SMK
```

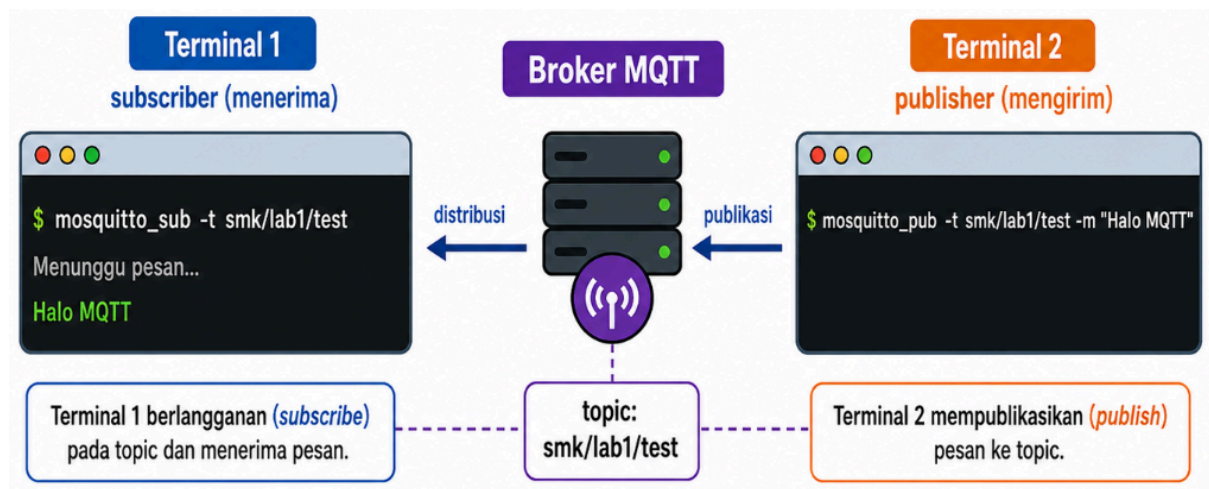
Tes dari laptop lain di jaringan yang sama:

```
mosquitto_sub -h 192.168.1.100 -t smk/lab1/test
```

Kirim pesan:

```
mosquitto_pub -h 192.168.1.100 -t smk/lab1/test -m "Tes dari laptop"
```

Mosquitto menyediakan utilitas `mosquitto_pub` dan `mosquitto_sub` untuk melakukan pengujian publish dan subscribe dari command line. ([GitHub](#))



Gambar 7.3 Broker MQTT harus diuji dari terminal sebelum digunakan oleh WeMOS mini D1 dan Android.

Target praktik:

Siswa harus melihat pesan muncul di terminal subscriber sebelum masuk ke WeMOS mini D1.

7.8 Username, Password, dan Larangan Broker Terbuka

Broker MQTT tidak boleh dibiarkan terbuka tanpa kontrol. Jika broker terbuka, orang lain di jaringan dapat membaca topic atau mengirim perintah palsu.

Konfigurasi sederhana:

```
sudo mosquitto_passwd -c /etc/mosquitto/passwd smk_iot
```

Buat file konfigurasi:

```
sudo nano /etc/mosquitto/conf.d/iot-smk.conf
```

Isi contoh:

```
listener 1883
allow_anonymous false
password_file /etc/mosquitto/passwd
```

Restart Mosquitto:

```
sudo systemctl restart mosquitto
```

Tes subscribe dengan user dan password:

```
mosquitto_sub -h 192.168.1.100 -t smk/lab1/test -u smk_iot -P
password_anda
```

Tes publish:

```
mosquitto_pub -h 192.168.1.100 -t smk/lab1/test -m "Aman
dengan password" -u smk_iot -P password_anda
```

Catatan keamanan:

Jangan menulis password asli di buku laporan. Gunakan contoh atau sensor:

```
password_anda
```

Ubuntu Server menyediakan **ufw** sebagai alat firewall yang sederhana untuk mengatur lalu lintas jaringan dan membatasi akses host. ([Ubuntu](#)) Untuk lab lokal, port 1883 hanya perlu dibuka di jaringan sekolah, bukan ke internet.

Aturan keras:

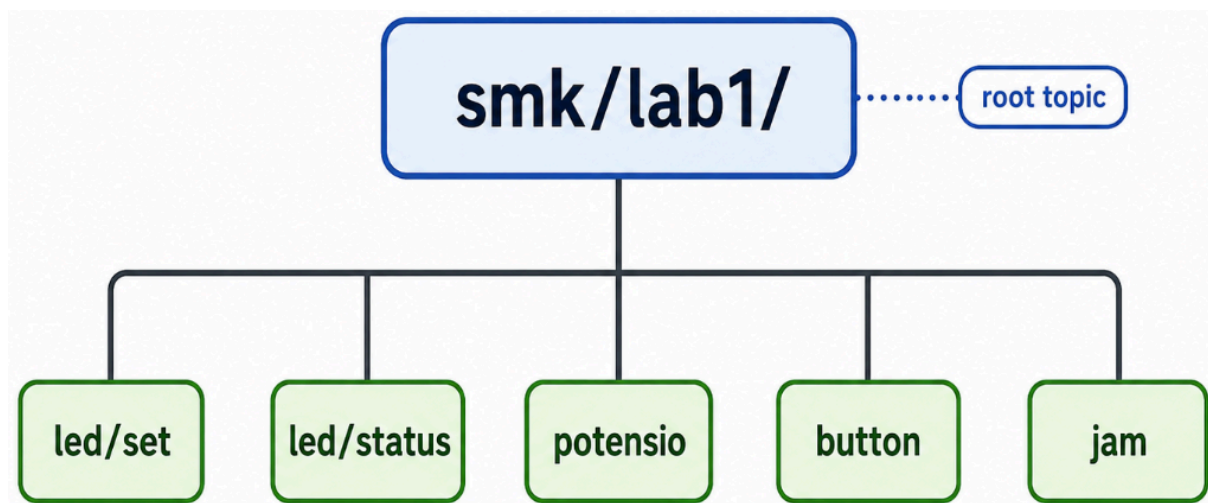
Broker MQTT terbuka adalah risiko. Broker untuk lab harus memakai username dan password.

7.9 Struktur Topic untuk Lab SMK

Struktur topic harus rapi sejak awal. Topic yang acak akan menyulitkan debugging.

Format yang disarankan:

```
smk/lab1/perangkat/data  
smk/lab1/perangkat/perintah  
smk/lab1/perangkat/status
```



Gambar 7.4 Struktur topic yang rapi membuat data IoT mudah dicari, diuji, dan dikembangkan.

Contoh topic buku:

```
smk/lab1/led/set  
smk/lab1/led/status  
smk/lab1/potensio  
smk/lab1/button  
smk/lab1/jam
```

Makna topic:

Topic	Fungsi
smk/lab1/led/set	Perintah ON/OFF ke LED
smk/lab1/led/status	Status LED dari WeMOS
smk/lab1/potensio	Nilai potensiometer
smk/lab1/button	Status tombol
smk/lab1/jam	Data jam digital
smk/lab1/test	Topic latihan

Contoh payload:

Topic	Payload
smk/lab1/led/set	ON / OFF
smk/lab1/led/status	ON / OFF
smk/lab1/potensio	0 sampai 1023
smk/lab1/button	PRESSED / RELEASED
smk/lab1/jam	00:05:21

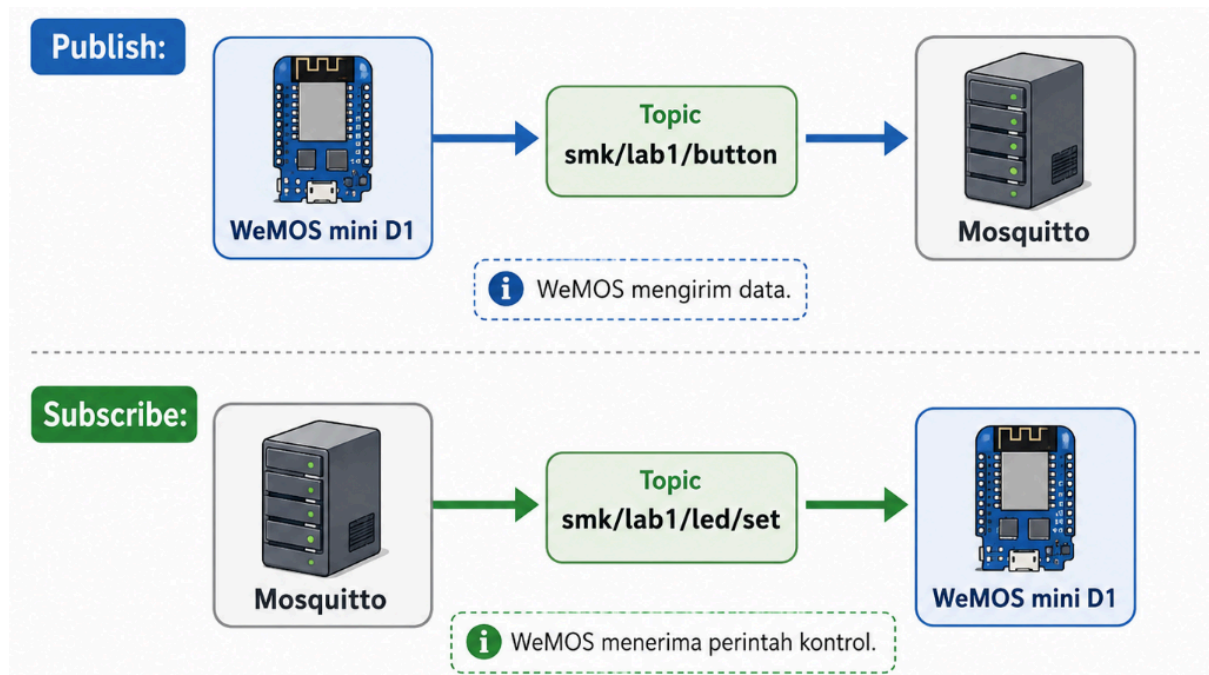
Prinsip topic:

Gunakan nama yang pendek, jelas, konsisten, dan mudah ditebak.

7.10 Praktik 2 — WeMOS mini D1 Publish Data Sederhana

Praktik ini mengirim data dari WeMOS mini D1 ke MQTT Broker. Data awal cukup sederhana: teks “online” atau angka potensiometer.

Library yang umum dipakai adalah **PubSubClient**, library MQTT client untuk perangkat berbasis Arduino. Untuk ESP8266, library ini dapat bekerja bersama [ESP8266WiFi](#).



Gambar 7.5 WeMOS mini D1 dapat mengirim data sensor sekaligus menerima perintah kontrol melalui MQTT.

Kode publish sederhana

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

const char* ssid = "Lab-IoT-SMK";
const char* password = "password_wifi_lab";

const char* mqtt_server = "192.168.1.100";
const char* mqtt_user = "smk_iot";
const char* mqtt_pass = "password_anda";

WiFiClient espClient;
PubSubClient client(espClient);

void setup_wifi() {
    Serial.print("Menghubungkan ke Wi-Fi");
```

```

WiFi.begin(ssid, password);

while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
}

Serial.println();
Serial.print("IP WeMOS: ");
Serial.println(WiFi.localIP());
}

void reconnect() {
    while (!client.connected()) {
        Serial.print("Menghubungkan ke MQTT...");

        if (client.connect("wemos_lab1", mqtt_user, mqtt_pass)) {
            Serial.println("terhubung");
        } else {
            Serial.print("gagal, rc=");
            Serial.print(client.state());
            Serial.println(" coba lagi 5 detik");
            delay(5000);
        }
    }
}

void setup() {
    Serial.begin(9600);
    setup_wifi();

    client.setServer(mqtt_server, 1883);
}

void loop() {
    if (!client.connected()) {
        reconnect();
    }

    client.loop();

    client.publish("smk/lab1/button", "online");
    Serial.println("Publish: smk/lab1/button = online");

    delay(3000);
}

```

Uji dari laptop

```
mosquitto_sub -h 192.168.1.100 -t smk/lab1/button -u smk_iot  
-P password_anda
```

Hasil:

```
online  
online  
online
```

Kalimat penting:

Jika laptop dapat membaca pesan dari WeMOS, berarti Wi-Fi, broker, topic, dan autentikasi sudah bekerja.

7.11 Praktik 3 — WeMOS mini D1 Subscribe Perintah ON/OFF

Praktik ini membuat WeMOS mini D1 menerima perintah dari broker untuk mengontrol LED.

Topic perintah:

```
smk/lab1/led/set
```

Payload:

```
ON  
OFF
```

Kode subscribe ON/OFF

```
#include <ESP8266WiFi.h>  
#include <PubSubClient.h>  
  
const char* ssid = "Lab-IoT-SMK";  
const char* password = "password_wifi_lab";  
  
const char* mqtt_server = "192.168.1.100";  
const char* mqtt_user = "smk_iot";  
const char* mqtt_pass = "password_anda";  
  
int ledPin = D5;  
  
WiFiClient espClient;  
PubSubClient client(espClient);  
  
void callback(char* topic, byte* payload, unsigned int  
length) {  
    String pesan = "";  
  
    for (unsigned int i = 0; i < length; i++) {  
        pesan += (char)payload[i];  
    }  
  
    Serial.print("Topic: ");  
    Serial.println(topic);  
    Serial.print("Pesan: ");  
    Serial.println(pesan);  
  
    if (pesan == "ON") {  
        digitalWrite(ledPin, HIGH);  
        client.publish("smk/lab1/led/status", "ON");  
    }  
  
    if (pesan == "OFF") {
```

```

        digitalWrite(ledPin, LOW);
        client.publish("smk/lab1/led/status", "OFF");
    }
}

void setup_wifi() {
    WiFi.begin(ssid, password);

    Serial.print("Menghubungkan Wi-Fi");
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }

    Serial.println();
    Serial.print("IP WeMOS: ");
    Serial.println(WiFi.localIP());
}

void reconnect() {
    while (!client.connected()) {
        Serial.print("Menghubungkan MQTT...");

        if (client.connect("wemos_led_lab1", mqtt_user,
mqtt_pass)) {
            Serial.println("terhubung");
            client.subscribe("smk/lab1/led/set");
        } else {
            Serial.print("gagal, rc=");
            Serial.println(client.state());
            delay(5000);
        }
    }
}

void setup() {
    Serial.begin(9600);

    pinMode(ledPin, OUTPUT);
    digitalWrite(ledPin, LOW);

    setup_wifi();

    client.setServer(mqtt_server, 1883);
    client.setCallback(callback);
}

void loop() {

```

```
    if (!client.connected()) {  
        reconnect();  
    }  
  
    client.loop();  
}
```

Kirim perintah dari laptop

Nyalakan LED:

```
mosquitto_pub -h 192.168.1.100 -t smk/lab1/led/set -m "ON" -u  
smk_iot -P password_anda
```

Matikan LED:

```
mosquitto_pub -h 192.168.1.100 -t smk/lab1/led/set -m "OFF"  
-u smk_iot -P password_anda
```

Pantau status:

```
mosquitto_sub -h 192.168.1.100 -t smk/lab1/led/status -u  
smk_iot -P password_anda
```

Target praktik:

Siswa melihat bahwa perintah dari laptop dapat mengubah kondisi LED melalui MQTT.

7.12 Troubleshooting MQTT dan Checklist Hasil

Masalah 1 — Mosquitto tidak berjalan

Cek:

```
systemctl status mosquitto
```

Solusi:

```
sudo systemctl restart mosquitto
```

Masalah 2 — Port 1883 tertutup

Cek:

```
ss -tulpn | grep 1883
```

Jika memakai firewall, izinkan port di jaringan lokal:

```
sudo ufw allow 1883/tcp
```

Masalah 3 — Salah username atau password

Gejala:

- WeMOS gagal konek MQTT.
- `client.state()` menunjukkan error.
- `mosquitto_pub` dari laptop gagal.

Solusi:

- Cek username.
- Cek password.
- Buat ulang password dengan `mosquitto_passwd`.
- Restart Mosquitto.

Masalah 4 — IP server salah

Gejala:

- WeMOS Wi-Fi tersambung, tetapi MQTT gagal.
- Laptop bisa internet, tetapi tidak bisa broker.

Solusi:

```
Cek IP server:  
ip addr
```

- Pastikan `mqtt_server` di kode sesuai IP server.

Masalah 5 — Topic salah

Gejala:

- Publish berhasil, tetapi subscriber tidak menerima.
- LED tidak merespons.

Solusi:

- Samakan topic persis.
- Cek huruf besar/kecil.
- Gunakan satu daftar topic resmi lab.

Checklist hasil Bab 7

Siswa dinyatakan berhasil jika mampu:

- Menjelaskan peran Ubuntu Server dalam IoT.
- Menjelaskan broker, publisher, subscriber, topic, payload.
- Menginstal Mosquitto.
- Mengecek status service Mosquitto.
- Menguji `mosquitto_pub` dan `mosquitto_sub`.
- Membuat username dan password MQTT.
- Menyusun struktur topic lab.
- Membuat WeMOS mini D1 publish pesan.
- Membuat WeMOS mini D1 subscribe perintah.
- Menyalakan dan mematikan LED lewat MQTT.

Latihan Bab 7

1. Apa fungsi Ubuntu Server dalam sistem IoT?
2. Mengapa server lokal cocok untuk lab SMK?
3. Mengapa server sebaiknya memakai IP statis?
4. Apa fungsi SSH?
5. Apa fungsi Mosquitto?
6. Jelaskan perbedaan broker, publisher, dan subscriber.
7. Apa perbedaan topic dan payload?
8. Mengapa broker MQTT tidak boleh terbuka tanpa password?
9. Apa fungsi `mosquitto_sub`?
10. Apa fungsi `mosquitto_pub`?
11. Mengapa topic harus rapi dan konsisten?
12. Apa yang harus dicek jika WeMOS tidak bisa terhubung ke MQTT?

Tugas Praktik Bab 7

Buat sistem MQTT lokal sederhana:

1. Ubuntu Server menjalankan Mosquitto.
2. Laptop subscribe topic:
`smk/lab1/test`
3. Laptop publish pesan:
`Halo MQTT dari SMK`
4. WeMOS mini D1 publish pesan:
`smk/lab1/button = online`
5. WeMOS mini D1 subscribe:
`smk/lab1/led/set`
6. Laptop mengirim:
`ON`
`OFF`
7. LED WeMOS menyala dan mati sesuai perintah.

Laporan singkat berisi:

- IP Ubuntu Server.
- Status service Mosquitto.
- Perintah `mosquitto_sub`.
- Perintah `mosquitto_pub`.
- Struktur topic.
- Screenshot terminal.
- Screenshot Serial Monitor.
- Foto LED menyala.
- Masalah yang muncul.
- Solusi yang dilakukan.

Target akhir Bab 7:

Siswa mampu menjelaskan dan membangun komunikasi MQTT lokal: **Ubuntu Server menjalankan Mosquitto, laptop menguji publish/subscribe, WeMOS mini D1 mengirim data, dan WeMOS mini D1 menerima perintah ON/OFF melalui topic yang rapi dan aman.**

BAB 8 — Android Dashboard untuk Monitoring dan Kontrol

8.1 Android sebagai Antarmuka IoT

Setelah WeMOS mini D1 dapat tersambung Wi-Fi dan berkomunikasi dengan Mosquitto MQTT Broker, siswa perlu melihat hasilnya dalam bentuk yang mudah dipahami. Di sinilah **Android dashboard** menjadi penting.

Android berfungsi sebagai **antarmuka manusia**. Data yang sebelumnya hanya muncul di terminal atau Serial Monitor dapat ditampilkan sebagai angka, tombol, status, dan indikator.

Contoh fungsi Android dalam proyek ini:

- Menampilkan nilai **potensiometer**.
- Menampilkan status **button**.
- Menampilkan data **jam digital**.
- Mengirim perintah **ON/OFF** untuk LED.
- Membantu siswa melihat alur MQTT secara nyata.

Dalam arsitektur buku:

WeMOS mini D1 → Wi-Fi → Mosquitto MQTT Broker → Android Dashboard
Android Dashboard → Mosquitto MQTT Broker → WeMOS mini D1

Kalimat penting:

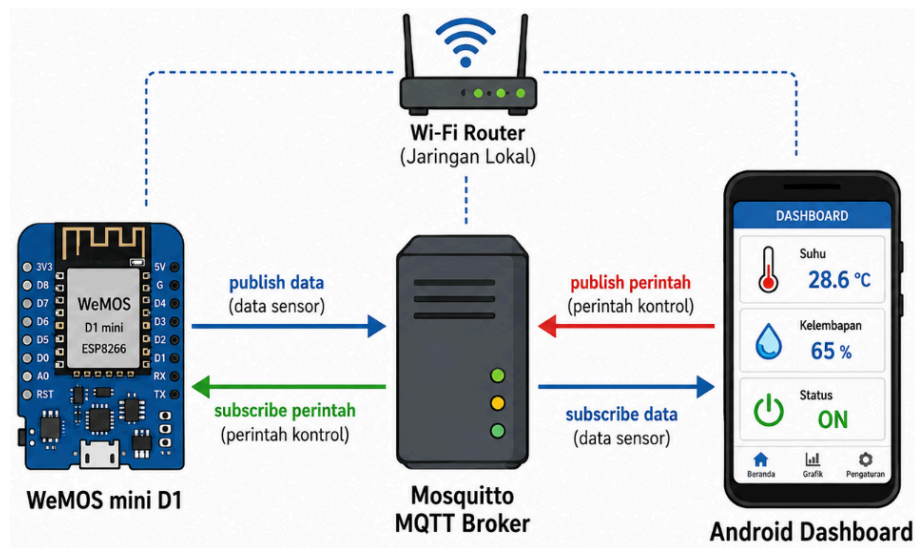
Android membuat data IoT terlihat dan kontrol IoT terasa nyata.

8.2 Konsep Dashboard MQTT

Dashboard MQTT bekerja dengan dua cara:

- **Subscribe** untuk membaca data.
- **Publish** untuk mengirim perintah.

MQTT adalah standar OASIS untuk protokol pesan IoT yang ringan dan memakai pola *publish/subscribe*. Pola ini cocok untuk perangkat kecil dan jaringan dengan bandwidth terbatas. ([MQTT](#))



Gambar 8.1 Android dashboard membaca data dari WeMOS mini D1 dan mengirim perintah kontrol melalui MQTT Broker.

Contoh monitoring:

```
WeMOS publish → smk/lab1/potensio → Mosquitto → Android  
subscribe
```

Contoh kontrol:

```
Android publish → smk/lab1/led/set → Mosquitto → WeMOS  
subscribe
```

Dashboard tidak berbicara langsung ke pin WeMOS. Dashboard berbicara ke **topic MQTT**. WeMOS membaca topic tersebut, lalu menjalankan aksi.

Prinsip penting:

Android tidak mengontrol LED secara langsung; Android mengirim payload ke topic, lalu WeMOS menjalankan perintah.

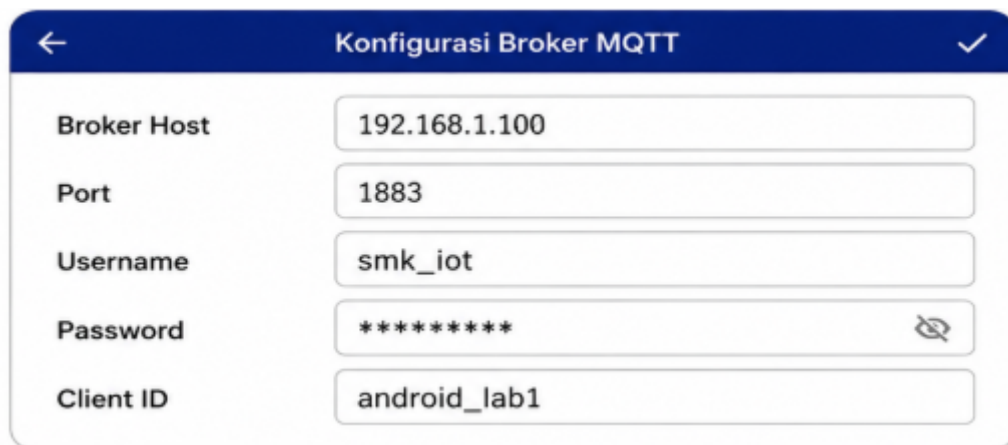
8.3 Konfigurasi Broker MQTT di Android

Aplikasi dashboard MQTT perlu mengetahui lokasi broker.

Data yang harus disiapkan:

Konfigurasi	Contoh
Broker / host	192.168.1.100
Port	1883
Username	smk_iot
Password	password_anda
Client ID	android_lab1
Protocol	MQTT TCP

Mosquitto adalah broker MQTT open source yang mendukung MQTT versi 5.0, 3.1.1, dan 3.1. Mosquitto ringan dan cocok dipakai dari perangkat kecil sampai server penuh. ([Eclipse Mosquitto](#))



Gambar 8.2 Konfigurasi broker yang benar menentukan apakah Android dapat terhubung ke Mosquitto MQTT Broker.

Langkah konsep konfigurasi:

1. Android tersambung ke Wi-Fi lab yang sama.
2. Masukkan IP Ubuntu Server.
3. Masukkan port MQTT.
4. Masukkan username dan password.
5. Simpan koneksi.
6. Tes koneksi.
7. Buat widget sesuai topic.

Aturan keamanan:

Jangan menulis password asli di laporan publik. Gunakan contoh atau samarkan.

8.4 Widget Monitoring Data

Widget monitoring dipakai untuk menampilkan data yang dikirim WeMOS mini D1.

Contoh topic monitoring:

```
smk/lab1/potensio  
smk/lab1/button  
smk/lab1/jam  
smk/lab1/led/status
```

Contoh payload:

Topic	Payload	Tampilan Android
smk/lab1/potensio	735	Nilai potencio: 735
smk/lab1/button	PRESSED	Tombol ditekan
smk/lab1/jam	00:12:35	Jam: 00:12:35
smk/lab1/led/status	ON	LED menyala

Widget yang disarankan:

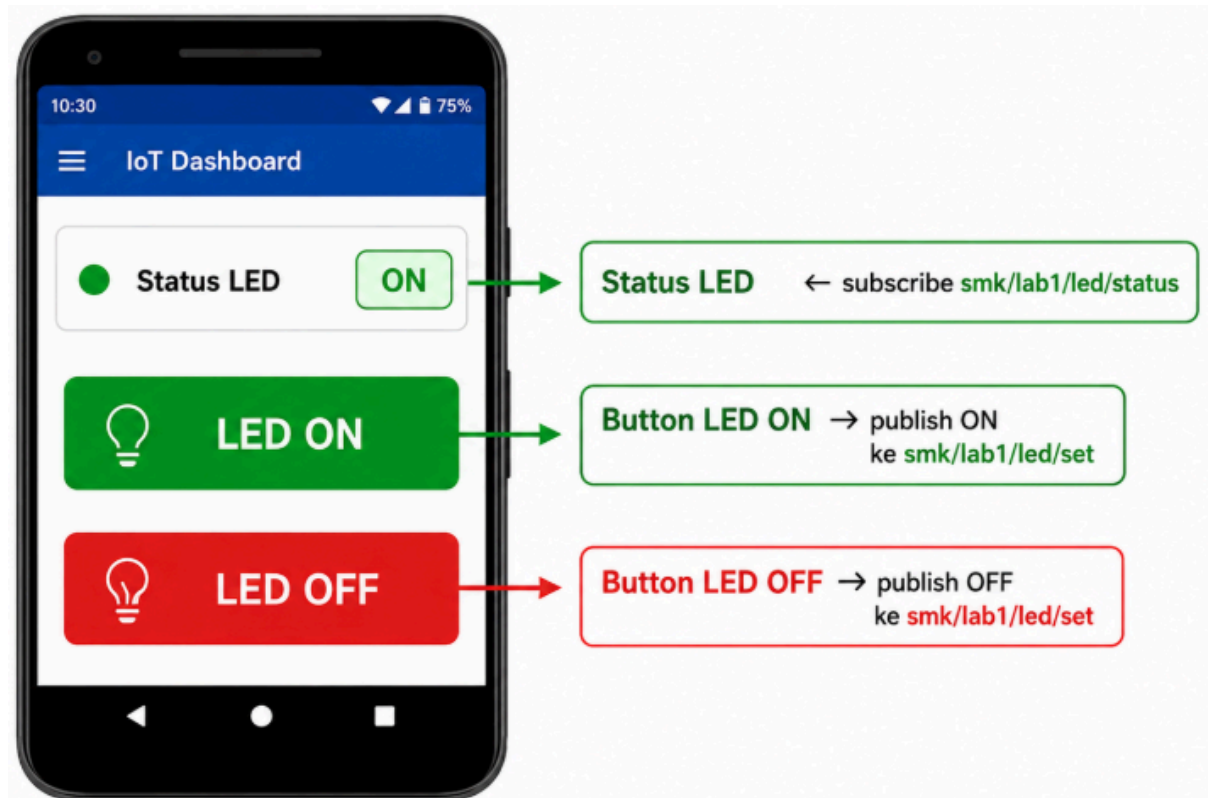
- *Text widget* untuk nilai potensiometer.
- *Text widget* untuk status button.
- *Text widget* untuk jam digital.
- *Indicator widget* untuk status LED.
- *Button widget* untuk perintah ON/OFF.

Prinsip dashboard:

Satu widget harus jelas membaca atau menulis topic apa.

8.5 Tombol ON/OFF LED

Tombol ON/OFF adalah praktik kontrol paling sederhana.



Gambar 8.3 Widget LED memungkinkan Android mengirim perintah ON/OFF dan membaca status LED melalui MQTT.

Topic:

`smk/lab1/led/set`

Payload:

ON
OFF

Alur kerja:

Android tombol ON
↓
Publish payload ON ke smk/lab1/led/set
↓
Mosquitto meneruskan pesan
↓
WeMOS mini D1 menerima pesan
↓
LED menyala
↓
WeMOS publish status ON ke smk/lab1/led/status
↓
Android menampilkan LED ON

Konfigurasi tombol ON:

Item	Isi
Widget	Button
Label	LED ON
Topic publish	smk/lab1/led/set
Payload	ON

Konfigurasi tombol OFF:

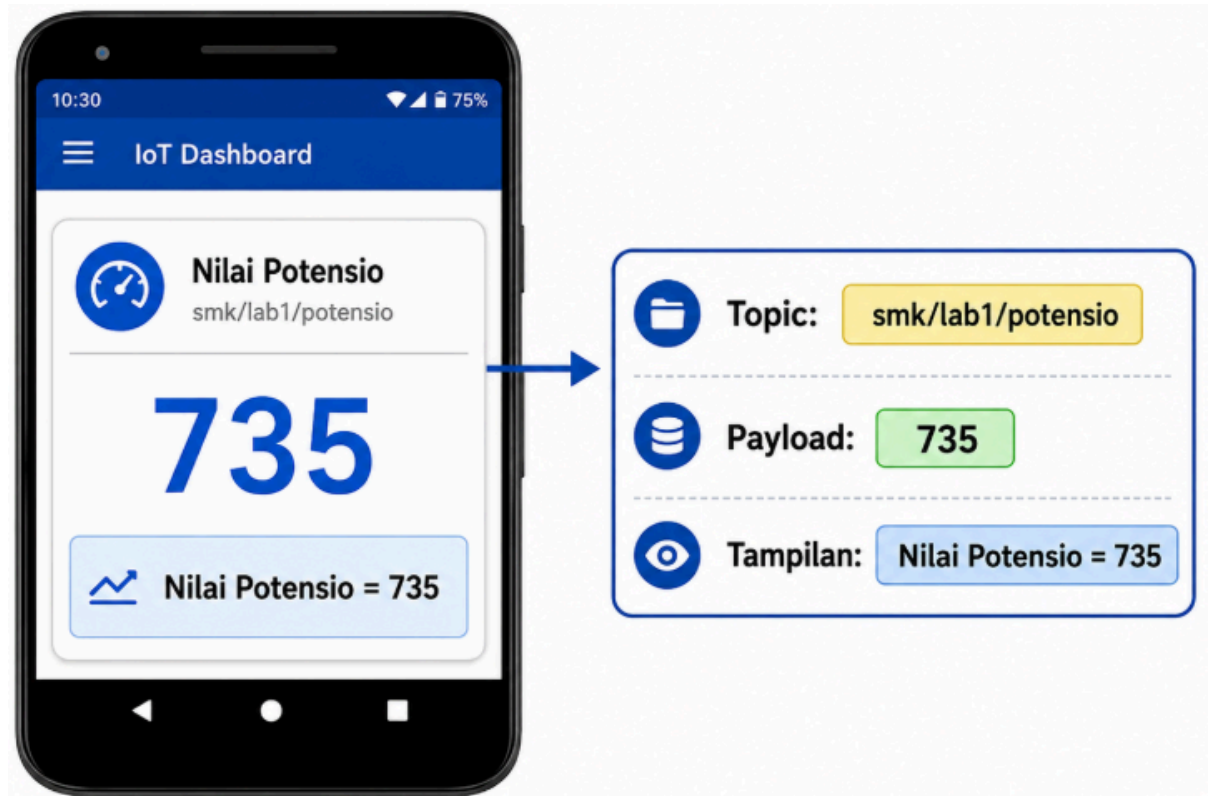
Item	Isi
Widget	Button
Label	LED OFF
Topic publish	smk/lab1/led/set
Payload	OFF

Catatan kritis:

Jika LED tidak merespons, jangan langsung menyalahkan Android. Cek topic, payload, koneksi broker, dan kode subscribe WeMOS.

8.6 Tampilan Nilai Potensiometer

Potensiometer mengirim nilai analog dari A0. Nilai ini bisa ditampilkan sebagai angka di dashboard.



Gambar 8.4 Nilai potensiometer dari WeMOS mini D1 dapat tampil langsung di Android dashboard melalui topic MQTT.

Topic:

```
smk/lab1/potensio
```

Payload contoh:

```
0
128
512
735
1023
```

Alur kerja:

```
Potensiometer diputar
↓
WeMOS membaca A0
↓
WeMOS publish nilai ke smk/lab1/potensio
↓
Android subscribe topic
↓
Nilai tampil di dashboard
```

Konfigurasi widget:

Item	Isi
Widget	Text / Value
Label	Nilai Potensio
Topic subscribe	smk/lab1/potensio
Format tampilan	Potensio: {payload}

Kalimat penting:

Potensiometer membuat siswa melihat hubungan antara gerakan fisik dan data digital di HP.

8.7 Tampilan Status Button

Button mengirim status digital. Statusnya sederhana:

```
PRESSED  
RELEASED
```

Topic:

```
smk/lab1/button
```

Contoh alur:

```
Button ditekan  
↓  
WeMOS membaca D6  
↓  
WeMOS publish PRESSED  
↓  
Android menampilkan "Tombol ditekan"
```

Konfigurasi widget:

Item	Isi
Widget	Text / Indicator
Label	Status Button
Topic subscribe	<code>smk/lab1/button</code>
Payload	<code>PRESSED / RELEASED</code>

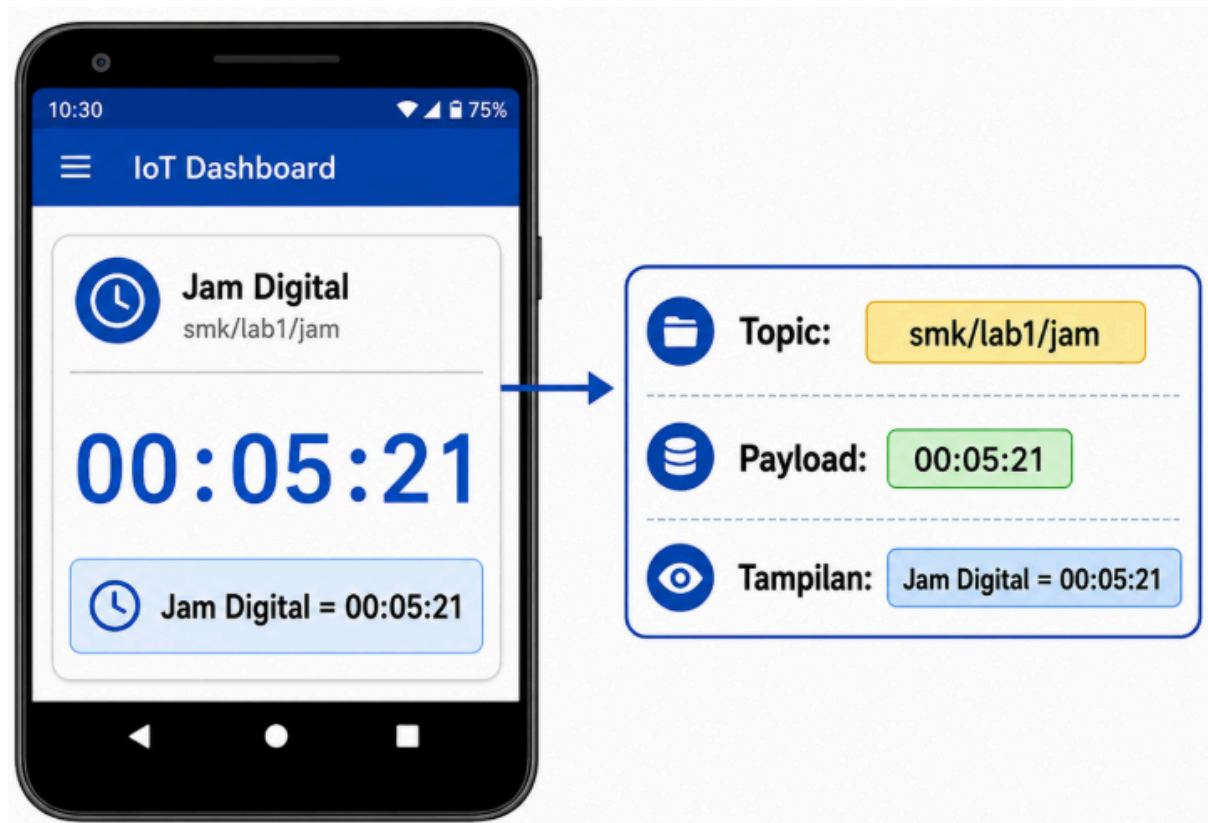
Catatan teknis:

Jika memakai `INPUT_PULLUP`, tombol ditekan biasanya terbaca **LOW** di kode. Namun payload MQTT tetap boleh dibuat lebih mudah dibaca manusia, misalnya `PRESSED`.

Prinsip tampilan:

Payload untuk dashboard sebaiknya manusiawi, bukan sekadar HIGH atau LOW.

8.8 Tampilan Jam Digital dari Topic MQTT



Gambar 8.5 Data jam digital dari WeMOS mini D1 dapat ditampilkan di Android sebagai hasil subscribe topic MQTT.

Bab sebelumnya sudah membuat jam digital sederhana dengan format:

HH:MM:SS

Di Bab 8, data jam dikirim ke MQTT agar tampil di Android.

Topic:

smk/lab1/jam

Payload contoh:

00:00:01
00:00:02
00:00:03

Alur kerja:

```
millis()  
  ↓  
format HH:MM:SS  
  ↓  
WeMOS publish ke smk/lab1/jam  
  ↓  
Android subscribe  
  ↓  
Jam tampil di dashboard
```

Konfigurasi widget:

Item	Isi
Widget	Text / Clock display
Label	Jam Digital
Topic subscribe	smk/lab1/jam
Format tampilan	Jam: {payload}

Catatan kritis:

Jam berbasis `millis()` bukan waktu nyata. Jika board di-reset, jam mulai dari nol. Untuk waktu nyata, pengembangan dapat memakai RTC atau NTP.

8.9 Payload Teks vs JSON Sederhana

Pada tahap awal, payload teks lebih mudah.

Contoh payload teks:

```
ON
OFF
PRESSED
RELEASED
735
00:05:21
```

Payload teks cocok untuk:

- Pemula.
- Praktik cepat.
- Debugging.
- LED ON/OFF.
- Status tombol.
- Jam sederhana.

Namun untuk data gabungan, JSON lebih rapi.

Contoh payload JSON:

```
{
  "potensio": 735,
  "button": "PRESSED",
  "jam": "00:05:21",
  "led": "ON"
}
```

Payload JSON cocok untuk:

- Data lebih dari satu nilai.
- Dashboard yang lebih kompleks.
- Integrasi ke Node-RED.
- Penyimpanan data.
- Pengembangan ke sistem monitoring.

Rekomendasi Bab 8:

Gunakan payload teks dulu. Kenalkan JSON sebagai pengembangan.

8.10 Kesalahan Umum dan Checklist Hasil

Kesalahan umum konfigurasi Android dashboard

1. **Android tidak satu jaringan dengan broker**
 - Solusi: pastikan Android tersambung ke Wi-Fi lab yang sama.
2. **IP broker salah**
 - Solusi: cek IP Ubuntu Server.
3. **Port salah**
 - Solusi: gunakan port MQTT yang benar, biasanya 1883.
4. **Username/password salah**
 - Solusi: uji dulu dengan `mosquitto_pub` dan `mosquitto_sub`.
5. **Topic berbeda satu huruf**
 - Solusi: salin topic dari daftar resmi lab.
6. **Payload tidak sesuai kode**
 - Solusi: jika kode menunggu `ON`, jangan kirim `on` atau `On`.
7. **WeMOS belum subscribe**
 - Solusi: cek Serial Monitor WeMOS.
8. **Broker Mosquitto mati**
 - Solusi: cek service di Ubuntu Server.
9. **Firewall menutup koneksi**
 - Solusi: pastikan port MQTT bisa diakses dari jaringan lokal.
10. **Android memakai data seluler, bukan Wi-Fi lab**
 - Solusi: gunakan Wi-Fi lab saat praktek.

Checklist hasil Bab 8

Siswa dinyatakan berhasil jika mampu:

- Menjelaskan fungsi Android dashboard.
- Menghubungkan Android ke broker MQTT.
- Membuat widget subscribe potensiometer.
- Membuat widget status button.
- Membuat widget jam digital.
- Membuat tombol publish ON/OFF.
- Menyalakan LED dari Android.
- Menjelaskan beda topic dan payload.
- Menjelaskan beda payload teks dan JSON.
- Melakukan troubleshooting saat dashboard tidak menerima data.

Latihan Bab 8

1. Apa fungsi Android dashboard dalam sistem IoT?
2. Apa perbedaan *publish* dan *subscribe* pada dashboard?
3. Mengapa Android harus satu jaringan dengan broker?
4. Apa fungsi IP broker?
5. Apa fungsi port MQTT?
6. Mengapa username dan password broker penting?
7. Apa topic untuk mengontrol LED?
8. Apa payload untuk menyalakan LED?
9. Apa topic untuk membaca potensiometer?
10. Mengapa payload teks cocok untuk pemula?
11. Kapan JSON lebih cocok digunakan?
12. Apa yang harus dicek jika Android tidak menerima data?

Tugas Praktik Bab 8

Buat dashboard Android sederhana dengan fitur:

1. **Nilai potensiometer**
 - Subscribe:
smk/lab1/potensio
2. **Status button**
 - Subscribe:
smk/lab1/button
3. **Kontrol LED**
 - Publish:
smk/lab1/led/set
 - Payload:
ON
 - OFF
4. **Status LED**
 - Subscribe:
smk/lab1/led/status
5. **Jam digital**
 - Subscribe:
smk/lab1/jam

Laporan singkat berisi:

- IP broker.
- Port MQTT.
- Username yang dipakai.
- Daftar topic.
- Screenshot konfigurasi broker.
- Screenshot widget LED.
- Screenshot nilai potensiometer.
- Screenshot status button.
- Screenshot jam digital.
- Masalah konfigurasi.
- Solusi yang dilakukan.

Target akhir Bab 8:

Siswa mampu membuat Android dashboard yang dapat **membaca data WeMOS mini D1** dan **mengirim perintah kontrol LED** melalui MQTT Broker lokal secara rapi, aman, dan mudah diuji.

BAB 9 — Proyek Praktik IoT Sederhana untuk SMK

9.1 Tujuan Bab 9 dan Pola Kerja Proyek

Bab ini bukan lagi hanya belajar perintah, fungsi, atau potongan kode. Bab ini meminta siswa membuat **sistem kecil yang bekerja utuh**.

Setiap proyek harus menjawab lima pertanyaan:

- **Input apa yang dibaca?**
- **Output apa yang dikendalikan?**
- **Topic MQTT apa yang dipakai?**
- **Payload apa yang dikirim?**
- **Bagaimana hasilnya terlihat di Android?**

Pola kerja setiap proyek:

1. **Tentukan tujuan proyek.**
2. **Siapkan komponen.**
3. **Rangkai hardware.**
4. **Upload program ke WeMOS mini D1.**
5. **Cek Serial Monitor.**
6. **Uji MQTT dari laptop.**
7. **Uji Android dashboard.**
8. **Catat masalah dan solusi.**

Inti Bab 9:

Siswa tidak hanya membuat alat menyala, tetapi mampu menjelaskan alur data dari pin, kode, MQTT, hingga dashboard.

9.2 Persiapan Umum Lab IoT SMK

Sebelum memulai proyek, pastikan semua perangkat sudah siap.

Perangkat utama:

- **WeMOS mini D1 / LOLIN D1 mini.**
- Breadboard.
- Kabel jumper.
- LED.
- Resistor 220Ω atau 330Ω.
- Push button.
- Potensiometer.
- Kabel USB data.
- Laptop Ubuntu Linux.
- Ubuntu Server.
- Mosquitto MQTT Broker.
- Android dashboard.

LOLIN D1 mini bekerja pada **3.3V**, memiliki **11 digital I/O**, **1 input analog A0 dengan batas 3.2V maksimum**, flash 4 MB, dan berbasis ESP8266EX; batas ini penting karena proyek Bab 9 memakai D5 untuk LED, D6 untuk button, dan A0 untuk potensiometer. (wemos.cc)

Checklist awal:

- Ubuntu Server sudah menyala.
- Mosquitto sudah aktif.
- IP broker diketahui.
- Username dan password MQTT siap.
- Android berada di jaringan Wi-Fi yang sama.
- WeMOS mini D1 sudah bisa tersambung Wi-Fi.
- Kabel USB adalah kabel data.
- Semua GND tersambung benar.

Aturan keselamatan:

Jangan menghubungkan beban listrik besar. Semua proyek Bab 9 cukup memakai LED, button, dan potensiometer.

9.3 Standar Topic MQTT Proyek

Topic harus konsisten agar siswa tidak bingung.

Gunakan root topic:

`smk/lab1/`

Daftar topic utama:

Fungsi	Topic	Payload
Perintah LED	<code>smk/lab1/led/set</code>	ON / OFF
Status LED	<code>smk/lab1/led/status</code>	ON / OFF
Status button	<code>smk/lab1/button</code>	PRESSED / RELEASED
Nilai potensiometer	<code>smk/lab1/potensio</code>	0 s.d. 1023
Jam digital	<code>smk/lab1/jam</code>	HH:MM:SS
Status perangkat	<code>smk/lab1/status</code>	ONLINE / ERROR

MQTT cocok untuk proyek ini karena merupakan standar OASIS untuk *messaging protocol* IoT yang ringan dan memakai pola *publish/subscribe*, sehingga perangkat kecil dapat mengirim dan menerima pesan melalui broker. ([MQTT](#))

Prinsip topic:

Topic harus pendek, konsisten, mudah dibaca, dan tidak berubah-ubah di tengah proyek.

9.4 Proyek 1 — LED Control via Android

Tujuan

Siswa membuat LED eksternal yang dapat dinyalakan dan dimatikan dari Android melalui MQTT.

Konsep

Android tidak langsung mengontrol pin D5. Android mengirim payload **ON** atau **OFF** ke broker. WeMOS mini D1 melakukan *subscribe* ke topic tersebut, lalu mengubah kondisi LED.

Alur:

```
Android publish ON/OFF
    ↓
Topic smk/lab1/led/set
    ↓
Mosquitto MQTT Broker
    ↓
WeMOS mini D1 subscribe
    ↓
LED menyala / mati
```

Komponen

- WeMOS mini D1.
- LED.
- Resistor 220Ω / 330Ω .
- Breadboard.
- Kabel jumper.
- Android dashboard.
- Mosquitto MQTT Broker.

Sambungan

D5 → Resistor 220Ω → kaki panjang LED
kaki pendek LED → GND

Topic

Fungsi	Topic	Payload
Android kirim perintah	<code>smk/lab1/led/set</code>	ON / OFF
WeMOS kirim status	<code>smk/lab1/led/status</code>	ON / OFF

Kode Inti

```
void callback(char* topic, byte* payload, unsigned int
length) {
    String pesan = "";

    for (unsigned int i = 0; i < length; i++) {
        pesan += (char)payload[i];
    }

    if (pesan == "ON") {
        digitalWrite(D5, HIGH);
        client.publish("smk/lab1/led/status", "ON");
    }

    if (pesan == "OFF") {
        digitalWrite(D5, LOW);
        client.publish("smk/lab1/led/status", "OFF");
    }
}
```

PubSubClient adalah library MQTT untuk Arduino yang memungkinkan perangkat mengirim dan menerima pesan MQTT, sehingga cocok untuk contoh WeMOS mini D1 yang perlu *publish* status LED dan *subscribe* perintah ON/OFF. ([Dokumentasi Arduino](#))

Output yang Diharapkan

- Tombol **LED ON** di Android menyalakan LED.
- Tombol **LED OFF** di Android mematikan LED.
- Status LED tampil di Android.

9.5 Proyek 2 — Button Status Monitor

Tujuan

Siswa membuat tombol fisik yang statusnya tampil di Android dashboard.

Konsep

Button adalah input digital. Saat button ditekan, WeMOS mini D1 membaca perubahan pin, lalu mengirim payload ke MQTT.

Alur:

```
Button ditekan
    ↓
WeMOS membaca D6
    ↓
WeMOS publish status
    ↓
Topic smk/lab1/button
    ↓
Android menampilkan status
```

Komponen

- WeMOS mini D1.
- Push button.
- Breadboard.
- Kabel jumper.
- Resistor jika memakai *pull-down* eksternal.
- MQTT Broker.
- Android dashboard.

Sambungan dengan **INPUT_PULLUP**

D6 → Button → GND

Dalam mode **INPUT_PULLUP**, tombol ditekan biasanya terbaca **LOW**, sehingga kode perlu menerjemahkan **LOW** menjadi **PRESSED**.

Topic

Fungsi	Topic	Payload
Status button	smk/lab1/button	PRESSED / RELEASED

Kode Inti

```
int buttonPin = D6;
int statusLama = HIGH;

void loop() {
    client.loop();

    int statusBaru = digitalRead(buttonPin);

    if (statusBaru != statusLama) {
        if (statusBaru == LOW) {
            client.publish("smk/lab1/button", "PRESSED");
            Serial.println("Button: PRESSED");
        } else {
            client.publish("smk/lab1/button", "RELEASED");
            Serial.println("Button: RELEASED");
        }

        statusLama = statusBaru;
        delay(200);
    }
}
```

Output yang Diharapkan

- Saat tombol ditekan, Android menampilkan **PRESSED**.
- Saat tombol dilepas, Android menampilkan **RELEASED**.
- Serial Monitor menampilkan perubahan status.

Catatan kritis:

Jika status tombol berubah sendiri, kemungkinan pin mengambang, GND salah, atau mode input belum benar.

9.6 Proyek 3 — Monitoring Potensiometer

Tujuan

Siswa membaca nilai analog dari potensiometer dan menampilkannya di Android dashboard melalui MQTT.

Konsep

Potensiometer menghasilkan tegangan berubah di kaki tengah. WeMOS mini D1 membaca tegangan ini melalui **A0** memakai `analogRead(A0)`.

Alur:

```
Potensiometer diputar
↓
Tegangan A0 berubah
↓
WeMOS membaca analogRead(A0)
↓
WeMOS publish nilai
↓
Android menampilkan angka
```

Komponen

- WeMOS mini D1.
- Potensiometer.
- Breadboard.
- Kabel jumper.
- MQTT Broker.
- Android dashboard.

Sambungan

```
3.3V → kaki luar potensiometer
A0   → kaki tengah potensiometer
GND  → kaki luar potensiometer
```

Peringatan:

A0 harus dijaga sesuai batas board. Gunakan 3.3V untuk praktik ini.

Topic

Fungsi	Topic	Payload
Nilai potensiometer	smk/lab1/potensio	angka analog

Kode Inti

```
void loop() {  
    client.loop();  
  
    int nilaiPot = analogRead(A0);  
  
    char payload[10];  
    sprintf(payload, "%d", nilaiPot);  
  
    client.publish("smk/lab1/potensio", payload);  
  
    Serial.print("Potensio: ");  
    Serial.println(payload);  
  
    delay(1000);  
}
```

Output yang Diharapkan

- Nilai potensiometer tampil di Serial Monitor.
- Nilai yang sama tampil di Android.
- Saat potensiometer diputar, angka berubah.

Prinsip penting:

Potensiometer mengajarkan bahwa kondisi fisik dapat berubah menjadi angka digital dan dikirim ke dashboard.

9.7 Proyek 4 — Jam Digital Serial dan MQTT

Tujuan

Siswa membuat jam digital sederhana berbasis `millis()`, menampilkannya di Serial Monitor, lalu mengirimnya ke MQTT agar tampil di Android.

Konsep

Jam digital ini bukan jam nyata. Ini adalah penghitung waktu sejak board menyala atau sejak di-reset.

Alur:

```
millis()  
  ↓  
total detik  
  ↓  
format HH:MM:SS  
  ↓  
Serial Monitor  
  ↓  
MQTT topic smk/lab1/jam  
  ↓  
Android dashboard
```

Komponen

- WeMOS mini D1.
- Komputer Ubuntu Linux.
- Serial Monitor.
- MQTT Broker.
- Android dashboard.

Topic

Fungsi	Topic	Payload
Jam digital	<code>smk/lab1/jam</code>	<code>HH:MM:SS</code>

Kode Inti

```
unsigned long waktuAwal = 0;

void setup() {
  Serial.begin(9600);
  waktuAwal = millis();
}

void loop() {
  client.loop();

  unsigned long totalDetik = (millis() - waktuAwal) / 1000;

  int detik = totalDetik % 60;
  int menit = (totalDetik / 60) % 60;
  int jam = (totalDetik / 3600) % 24;

  char waktu[10];
  sprintf(waktu, "%02d:%02d:%02d", jam, menit, detik);

  Serial.println(waktu);
  client.publish("smk/lab1/jam", waktu);

  delay(1000);
}
```

Fungsi `millis()` mengembalikan jumlah milidetik sejak program mulai berjalan; karena itu cocok untuk latihan waktu relatif, tetapi bukan pengganti RTC atau NTP untuk waktu nyata. ([Dokumentasi Arduino](#))

Output yang Diharapkan

Serial Monitor menampilkan:

- 00:00:01
- 00:00:02
- 00:00:03

Android dashboard menampilkan:

- Jam Digital = 00:00:03

Catatan kritis:

Jam berbasis `millis()` akan mulai dari nol setelah board di-reset. Untuk waktu nyata, gunakan RTC atau NTP sebagai pengembangan.

9.8 Proyek 5 — Smart Mini Panel SMK

Tujuan

Siswa menggabungkan semua proyek menjadi satu panel kecil:

- Android mengontrol LED.
- Button mengirim status.
- Potensiometer mengirim nilai analog.
- Jam digital tampil di Android.
- Semua komunikasi memakai MQTT.

Konsep

Smart Mini Panel adalah gabungan dari empat fungsi:

```
LED control
Button monitor
Potensio monitor
Jam digital
```

Arsitektur:

```
LED / Button / Potensio / Jam
    ↓
WeMOS mini D1
    ↓
Wi-Fi
    ↓
Mosquitto MQTT Broker
    ↓
Android Dashboard
```

Komponen

- WeMOS mini D1.
- LED.
- Resistor 220Ω / 330Ω.
- Button.
- Potensiometer.
- Breadboard.
- Kabel jumper.
- Ubuntu Server Mosquitto.
- Android dashboard.

Sambungan

LED:

D5 → Resistor 220Ω → LED → GND

Button:

D6 → Button → GND

Potensiometer:

3.3V → kaki luar potensiometer

A0 → kaki tengah potensiometer

GND → kaki luar potensiometer

Topic Gabungan

Fungsi	Topic	Arah
LED command	<code>smk/lab1/led/set</code>	Android → WeMOS
LED status	<code>smk/lab1/led/status</code>	WeMOS → Android
Button	<code>smk/lab1/button</code>	WeMOS → Android
Potensiometer	<code>smk/lab1/potensio</code>	WeMOS → Android
Jam digital	<code>smk/lab1/jam</code>	WeMOS → Android

Output yang Diharapkan

Android dashboard menampilkan:

```
Nilai Potensio = 735
Status Button = PRESSED
Status LED = ON
Jam Digital = 00:05:21
```

Android juga memiliki tombol:

```
LED ON
LED OFF
```

Nilai strategis proyek:

Smart Mini Panel membuat siswa memahami sistem IoT utuh: input, output, jaringan, broker, dashboard, kontrol, monitoring, dan troubleshooting.

9.9 Checklist Pengujian dan Troubleshooting

Checklist Pengujian Umum

Komponen	Cek
WeMOS mini D1	Menyala dan bisa di-upload
Wi-Fi	Tersambung dan mendapat IP
MQTT Broker	Mosquitto aktif
Android	Satu jaringan dengan broker
LED	Menyala saat payload ON
Button	Mengirim PRESSED / RELEASED
Potensiometer	Nilai berubah saat diputar
Jam digital	Format HH:MM:SS tampil
Topic	Sama persis antara kode dan dashboard
Password MQTT	Benar dan tidak ditulis sembarangan

Troubleshooting Cepat

LED tidak menyala

- Cek D5.
- Cek resistor.
- Cek polaritas LED.
- Cek topic `smk/lab1/led/set`.
- Cek payload `ON` / `OFF`.

Button tidak terbaca

- Cek D6.
- Cek GND.
- Cek `INPUT_PULLUP`.
- Ingat: ditekan biasanya terbaca `LOW`.

Potensiometer tidak berubah

- Cek kaki tengah ke A0.
- Cek 3.3V dan GND.
- Jangan pakai tegangan di luar batas A0.

Android tidak menerima data

- Cek IP broker.
- Cek port 1883.
- Cek username/password.
- Cek Android satu Wi-Fi dengan server.
- Cek topic subscribe.

MQTT gagal

- Cek service Mosquitto.
- Cek firewall.
- Cek IP server.
- Cek username/password.
- Uji dulu dengan `mosquitto_pub` dan `mosquitto_sub`.

Mosquitto adalah broker MQTT open source yang mendukung MQTT versi 5.0, 3.1.1, dan 3.1, sehingga cocok untuk pembelajaran broker lokal di Ubuntu Server. ([Eclipse Foundation](#))

Latihan Bab 9

1. Mengapa proyek IoT harus memakai topic yang rapi?
2. Apa beda `smk/lab1/led/set` dan `smk/lab1/led/status`?
3. Mengapa Android tidak langsung mengontrol pin D5?
4. Mengapa button dengan `INPUT_PULLUP` biasanya terbaca `LOW` saat ditekan?
5. Mengapa potensiometer harus masuk ke A0?
6. Mengapa nilai jam berbasis `millis()` bukan waktu nyata?
7. Apa fungsi Mosquitto dalam semua proyek Bab 9?
8. Mengapa WeMOS mini D1 bisa menjadi publisher dan subscriber sekaligus?
9. Apa yang harus dicek jika Android tidak menerima payload?
10. Mengapa laporan proyek harus mencatat masalah dan solusi?

Tugas Praktik Bab 9

Buat **Smart Mini Panel SMK** dengan fitur minimum:

- LED dikontrol dari Android.
- Button mengirim status ke Android.
- Potensiometer mengirim nilai analog ke Android.
- Jam digital tampil di Serial Monitor.
- Jam digital dikirim ke MQTT.
- Android menampilkan semua data.

Laporan berisi:

- Judul proyek.
- Tujuan proyek.
- Daftar komponen.
- Diagram sambungan.
- Daftar topic MQTT.
- Payload yang digunakan.
- Screenshot Serial Monitor.
- Screenshot Android dashboard.
- Foto breadboard.
- Hasil pengujian.
- Masalah yang ditemukan.
- Solusi yang dilakukan.
- Kesimpulan.

Target akhir Bab 9:

Siswa mampu membangun sistem IoT kecil yang lengkap: **membaca input, mengontrol output, mengirim data lewat MQTT, menerima perintah dari Android, dan mendokumentasikan proyek secara teknis.**

BAB 10 — Keamanan, Troubleshooting, Dokumentasi, dan Proyek Akhir

10.1 Mengapa Keamanan dan Dokumentasi Penting

Banyak siswa berhenti ketika LED sudah menyala atau dashboard sudah tampil. Itu belum cukup.

Dalam dunia teknis, proyek yang baik harus bisa:

- **Diuji ulang** oleh orang lain.
- **Diperbaiki** saat gagal.
- **Diamankan** agar tidak mudah disalahgunakan.
- **Dijelaskan** dengan diagram, topic, kode, dan hasil uji.
- **Dikembangkan** menjadi sistem yang lebih besar.

IoT menyentuh perangkat fisik. Jika salah tegangan, board bisa rusak. Jika salah topic, data tidak terbaca. Jika broker terbuka, orang lain dapat mengirim perintah palsu. Jika dokumentasi buruk, proyek sulit diperbaiki.

Prinsip akhir buku:

Siswa SMK harus berpikir seperti teknisi sistem, bukan hanya perakitan alat.

10.2 Checklist Keamanan Rangkaian

Sebelum menyalakan proyek akhir, siswa wajib memeriksa rangkaian.

Checklist daya dan tegangan:

- **WeMOS mini D1 memakai logika 3.3V.**
- Jangan memasukkan sinyal 5V langsung ke GPIO.
- Gunakan 3.3V untuk potensiometer ke A0.
- Perhatikan batas tegangan A0.
- Jangan mengambil arus besar dari pin WeMOS.
- Gunakan resistor 220Ω / 330Ω untuk LED.
- Satukan **GND** semua perangkat yang saling bertukar sinyal.

Checklist sambungan:

- LED: D5 → resistor → LED → GND.
- Button: D6 → button → GND, mode INPUT_PULLUP.
- Potensiometer: 3.3V → potensiometer → GND, kaki tengah ke A0.
- Kabel tidak longgar.
- Tidak ada hubungan pendek antara 3.3V/5V dan GND.
- Pin di kode sama dengan pin di rangkaian.

Checklist relay DC bila dipakai sebagai pengembangan:

- Gunakan beban DC rendah.
- Jangan memakai beban berbahaya tanpa guru.
- Pastikan relay module sesuai tegangan.
- Gunakan catu daya eksternal jika beban membutuhkan arus besar.
- GND catu daya eksternal dan GND WeMOS harus bersama.

Kalimat keras:

Kalau ragu, jangan nyalakan. Periksa ulang rangkaian.

10.3 Keamanan MQTT: Password, Topic, dan Akses

MQTT Broker adalah pusat pesan. Jika broker terbuka tanpa password, siapa pun di jaringan dapat mencoba membaca data atau mengirim perintah.

Mosquitto mendukung beberapa metode autentikasi, termasuk **password file**, *authentication plugin*, dan akses anonim. Dokumentasi Mosquitto juga menjelaskan bahwa jika `allow_anonymous` dibuat `false`, hanya user yang ada di `password_file` yang dapat terhubung. ([Eclipse Mosquitto](#)) ([Eclipse Mosquitto](#))

Konfigurasi aman minimum:

```
listener 1883
allow_anonymous false
password_file /etc/mosquitto/passwd
```

Aturan keamanan MQTT:

- Jangan memakai broker tanpa password.
- Jangan menulis password asli di laporan publik.
- Gunakan username khusus untuk lab.
- Gunakan topic yang rapi.
- Jangan memakai topic terlalu umum seperti `test` untuk proyek akhir.
- Jangan membuka port MQTT ke internet.
- Batasi akses hanya untuk jaringan lokal sekolah.

Contoh topic rapi:

```
smk/lab1/led/set
smk/lab1/led/status
smk/lab1/button
smk/lab1/potensio
smk/lab1/jam
```

Prinsip keamanan:

Topic rapi memudahkan debugging. Password mencegah akses sembarangan. Firewall membatasi jalur masuk.

10.4 Backup Konfigurasi Mosquitto dan Kode Proyek

Proyek yang tidak di-*backup* mudah hilang. Server bisa rusak. Konfigurasi bisa salah edit. Kode bisa tertimpa.

File penting yang perlu di-*backup*:

```
/etc/mosquitto/conf.d/iot-smk.conf
/etc/mosquitto/passwd
/opt/iot-smk/kode-wemos/
/opt/iot-smk/catatan-topic/
/opt/iot-smk/laporan/
```

Perintah backup sederhana:

```
sudo mkdir -p /opt/iot-smk/backup
sudo cp /etc/mosquitto/conf.d/iot-smk.conf
/opt/iot-smk/backup/
sudo cp /etc/mosquitto/passwd /opt/iot-smk/backup/
```

Backup folder proyek:

```
tar -czvf backup-iot-smk.tar.gz /opt/iot-smk/
```

Penjelasan singkat:

- `mkdir -p` membuat folder jika belum ada.
- `cp` menyalin file.
- `tar -czvf` membuat arsip terkompresi.
- `/opt/iot-smk/` menjadi folder proyek IoT sekolah.

Aturan dokumentasi:

Setiap perubahan konfigurasi harus dicatat: tanggal, file yang diubah, alasan, dan hasil uji.

10.5 Troubleshooting WeMOS mini D1

1. Gagal upload

Kemungkinan penyebab:

- Board salah pilih.
- Port salah.
- Kabel USB hanya untuk *charging*.
- Rangkaian eksternal mengganggu proses boot.
- Pin tertentu tertarik ke kondisi yang salah saat boot.

Solusi:

- Pilih board **LOLIN(WEMOS) D1 R2 & mini**.
- Cek port serial.
- Ganti kabel USB data.
- Lepas rangkaian eksternal sementara.
- Upload program paling sederhana dulu.

2. Port tidak muncul

Cek di Ubuntu Linux:

```
ls /dev/ttyUSB*  
ls /dev/ttyACM*
```

Solusi:

- Cabut-pasang kabel USB.
- Ganti kabel data.
- Cek chip USB serial.
- Coba port USB lain.

3. Wi-Fi gagal konek

Cek:

- SSID benar.
- Password benar.
- Router menyala.
- Jarak tidak terlalu jauh.
- Jaringan 2.4 GHz tersedia.
- Serial Monitor menampilkan status koneksi.

4. MQTT tidak tersambung

Cek:

- IP broker benar.
- Port 1883 terbuka.
- Username/password benar.
- Mosquitto berjalan.
- WeMOS dan server satu jaringan.

5. Sensor tidak terbaca

Cek:

- Pin sesuai kode.
- GND tersambung.
- Tegangan benar.
- A0 tidak menerima tegangan berlebih.
- Serial Monitor menampilkan nilai mentah.

Prinsip troubleshooting WeMOS:

Mulai dari daya, kabel, port, Wi-Fi, IP, MQTT, baru sensor.

10.6 Troubleshooting Ubuntu Server

Ubuntu Server adalah pusat layanan. Jika server bermasalah, Android dan WeMOS juga ikut gagal.

1. IP berubah

Cek IP:

```
ip addr
```

Solusi:

- Gunakan IP statis.
- Catat IP broker.
- Pastikan kode WeMOS memakai IP terbaru.
- Pastikan Android memakai host broker yang benar.

2. Mosquitto mati

Cek status:

```
systemctl status mosquitto
```

Restart:

```
sudo systemctl restart mosquitto
```

Aktifkan saat boot:

```
sudo systemctl enable mosquitto
```

3. Firewall menutup port

Ubuntu Server menggunakan **ufw** sebagai alat firewall yang lebih sederhana untuk mengatur aturan jaringan dan mengontrol lalu lintas host. ([Ubuntu](#))

Cek status:

```
sudo ufw status
```

Buka port MQTT di jaringan lokal:

```
sudo ufw allow 1883/tcp
```

4. Password salah

Buat ulang password:

```
sudo mosquitto_passwd /etc/mosquitto/passwd smk_iot  
sudo systemctl restart mosquitto
```

5. Broker tidak bisa diakses dari perangkat lain

Cek:

- Mosquitto mendengarkan port 1883.
- Konfigurasi listener benar.
- Firewall tidak menutup port.
- Server dan klien satu jaringan.
- Android tidak memakai jaringan seluler.

Prinsip server:

Jika broker mati, seluruh sistem MQTT ikut mati.

10.7 Troubleshooting Android Dashboard

Android dashboard sering gagal bukan karena sistem rusak, tetapi karena konfigurasi kecil salah.

Masalah umum:

- Android tidak satu Wi-Fi dengan Ubuntu Server.
- IP broker salah.
- Port salah.
- Username/password salah.
- Topic berbeda satu huruf.
- Payload tidak sama dengan kode.
- Broker belum berjalan.
- Firewall menutup port.
- Widget memakai mode *publish*, padahal perlu *subscribe*.
- Widget memakai *subscribe*, padahal harus *publish*.

Checklist Android:

Item	Contoh benar
Broker Host	192.168.1.100
Port	1883
Username	smk_iot
Password	disamarkan
Subscribe potensio	smk/lab1/potensio
Publish LED	smk/lab1/led/set
Payload LED ON	ON
Payload LED OFF	OFF
Subscribe jam	smk/lab1/jam

Tes cepat:

Sebelum menyalahkan Android, uji broker dari laptop:

```
mosquitto_sub -h 192.168.1.100 -t smk/lab1/# -u smk_iot -P  
password_anda
```

Lalu kirim:

```
mosquitto_pub -h 192.168.1.100 -t smk/lab1/led/set -m "ON" -u  
smk_iot -P password_anda
```

Kalimat penting:

Jika terminal bisa menerima data tetapi Android tidak, masalah kemungkinan ada di konfigurasi dashboard.

10.8 Format Laporan dan Presentasi Proyek SMK

Proyek akhir harus dilaporkan dengan rapi. Laporan bukan formalitas. Laporan adalah bukti bahwa siswa memahami sistem.

Format laporan proyek:

1. **Judul proyek**
2. **Nama siswa / kelompok**
3. **Tujuan proyek**
4. **Daftar komponen**
5. **Diagram rangkaian**
6. **Diagram arsitektur MQTT**
7. **Daftar topic dan payload**
8. **Kode program**
9. **Screenshot Serial Monitor**
10. **Screenshot Android dashboard**
11. **Foto rangkaian**
12. **Hasil pengujian**
13. **Masalah yang ditemukan**
14. **Solusi**
15. **Kesimpulan**
16. **Ide pengembangan**

Template laporan proyek memang disarankan dalam dokumen acuan agar siswa mencatat judul, tujuan, komponen, diagram, kode, hasil uji, foto, masalah, solusi, dan kesimpulan.

Format presentasi 5 menit:

- 1 menit: masalah dan tujuan.
- 1 menit: rangkaian dan komponen.
- 1 menit: topic MQTT dan alur data.
- 1 menit: demo Android dashboard.
- 1 menit: masalah, solusi, dan pengembangan.

Prinsip presentasi:

Jangan hanya demo alat. Jelaskan alur data.

10.9 Proyek Akhir — Smart IoT Learning Panel dengan WeMOS mini D1

Proyek akhir buku ini adalah:

Smart IoT Learning Panel dengan WeMOS mini D1

Tujuan:

Membuat panel pembelajaran IoT sederhana yang menggabungkan **LED**, **button**, **potensiometer**, **jam digital**, **MQTT**, **Ubuntu Server**, dan **Android dashboard**.

Fitur minimum:

- LED dapat dikontrol dari Android.
- Button mengirim status ke MQTT.
- Potensiometer mengirim nilai analog.
- Jam digital tampil di Serial Monitor.
- Jam digital dikirim ke MQTT.
- Android menampilkan semua data.
- Broker berjalan di Ubuntu Server lokal.
- Laporan berisi diagram, foto rangkaian, topic MQTT, kode, hasil uji, dan troubleshooting.

Fitur proyek akhir ini mengikuti arah dokumen attached yang menempatkan Smart IoT Learning Panel sebagai gabungan LED, button, potensiometer, jam digital, MQTT, dan Android dashboard.

Daftar topic proyek akhir:

Fungsi	Topic	Arah
Perintah LED	<code>smk/lab1/led/set</code>	Android → WeMOS
Status LED	<code>smk/lab1/led/status</code>	WeMOS → Android
Status button	<code>smk/lab1/button</code>	WeMOS → Android
Nilai potensiometer	<code>smk/lab1/potensio</code>	WeMOS → Android
Jam digital	<code>smk/lab1/jam</code>	WeMOS → Android

Kriteria lulus proyek akhir:

- Semua fitur berjalan.
- Topic konsisten.
- Android tampil benar.
- Serial Monitor menampilkan status.
- Mosquitto memakai username/password.
- Ada dokumentasi troubleshooting.
- Siswa mampu menjelaskan alur data dari board ke Android.

10.10 Ide Pengembangan Lanjutan

Setelah proyek akhir berhasil, siswa dapat mengembangkan sistem secara bertahap.

1. Node-RED

Node-RED dapat dipakai untuk membuat alur visual: MQTT masuk, diproses, lalu dikirim ke dashboard atau database. Cocok sebagai jembatan dari MQTT ke sistem otomasi yang lebih rapi.

2. InfluxDB

InfluxDB cocok untuk menyimpan data waktu seperti nilai potensiometer, suhu, cahaya, atau status sensor dari waktu ke waktu.

3. Grafana

Grafana dapat dipakai untuk membuat dashboard grafik berbasis data historis. Contoh stack IoT lokal dapat menggabungkan MQTT Broker, Node-RED, InfluxDB, dan Grafana untuk data logging dan visualisasi. ([Dokumentasi Arduino](#))

4. ESPHome

ESPHome mendukung ESP8266 dan ESP32 dengan konfigurasi YAML, serta menyediakan komponen sensor, switch, MQTT, dan banyak modul lain. Ini cocok sebagai jalur pengembangan setelah siswa memahami konsep manual lewat kode. ([ESPHome - Smart Home Made Simple](#))

5. Sensor suhu dan sensor cahaya

Setelah button dan potensiometer, siswa dapat menambah sensor suhu, kelembaban, cahaya, atau kualitas udara sederhana.

6. Relay DC

Relay DC dapat dipakai untuk mengontrol beban rendah seperti LED strip DC atau kipas kecil. Hindari beban berbahaya pada tahap pemula.

7. RTC

RTC cocok jika proyek membutuhkan jam yang tetap berjalan walau board mati.

8. NTP

NTP cocok jika board terhubung Wi-Fi dan membutuhkan waktu dari jaringan.

Penutup buku:

Siswa yang menyelesaikan proyek akhir ini sudah memahami dasar IoT utuh: rangkaian, input, output, Wi-Fi, server, MQTT, Android dashboard, keamanan, troubleshooting, dan dokumentasi.

Latihan Bab 10

1. Mengapa proyek IoT belum selesai hanya karena LED menyala?
2. Mengapa broker MQTT tidak boleh terbuka tanpa password?
3. Apa fungsi topic yang rapi?
4. Mengapa konfigurasi Mosquitto perlu di-*backup*?
5. Apa yang dicek jika port WeMOS tidak muncul?
6. Apa yang dicek jika Wi-Fi gagal tersambung?
7. Apa yang dicek jika MQTT tidak tersambung?
8. Apa yang dicek jika Android tidak menerima data?
9. Mengapa laporan proyek wajib memuat troubleshooting?
10. Apa manfaat Node-RED, InfluxDB, dan Grafana dalam pengembangan IoT?
11. Apa perbedaan RTC dan NTP?
12. Apa fitur minimum Smart IoT Learning Panel?

Tugas Praktik Bab 10

Buat dan presentasikan proyek akhir:

Smart IoT Learning Panel dengan WeMOS mini D1

Fitur wajib:

- LED dikontrol dari Android.
- Button mengirim status ke MQTT.
- Potensiometer mengirim nilai analog.
- Jam digital tampil di Serial Monitor.
- Jam digital dikirim ke MQTT.
- Android menampilkan semua data.
- Mosquitto berjalan di Ubuntu Server lokal.
- Broker memakai username dan password.
- Topic tersusun rapi.
- Ada laporan troubleshooting.

Output laporan:

- Diagram rangkaian.
- Diagram MQTT.
- Daftar topic dan payload.
- Kode program.
- Screenshot Serial Monitor.
- Screenshot Android dashboard.
- Foto rangkaian.
- Hasil uji.
- Masalah dan solusi.
- Kesimpulan.

Target akhir Bab 10:

Siswa mampu menyelesaikan proyek IoT sederhana secara profesional: **aman, terdokumentasi, dapat diuji, dapat diperbaiki, dan dapat dipresentasikan.**

Lampiran A — Daftar Komponen Praktik

Lampiran ini berisi komponen dasar untuk praktik IoT sederhana menggunakan **WeMOS mini D1 / LOLIN D1 mini**, Ubuntu Server, MQTT, dan Android dashboard.

Komponen	Fungsi
WeMOS mini D1 / LOLIN D1 mini	Board utama berbasis ESP8266 untuk membaca input, mengendalikan output, dan terhubung ke Wi-Fi.
Breadboard	Papan percobaan tanpa solder untuk merangkai LED, button, potensiometer, buzzer, dan modul lain.
Kabel jumper	Kabel penghubung antar pin WeMOS, breadboard, sensor, dan aktuator.
LED	Output digital sederhana untuk indikator ON/OFF.
Resistor	Membatasi arus, terutama pada LED. Gunakan 220Ω atau 330Ω untuk praktik dasar.
Push button	Input digital untuk membaca status ditekan atau dilepas.
Potensiometer	Input analog untuk menghasilkan nilai berubah pada pin A0.
Relay module DC	Saklar elektronik untuk mengontrol beban DC kecil.
Buzzer	Output suara sederhana untuk alarm atau indikator.
Kabel USB data	Untuk upload program dan komunikasi serial antara WeMOS mini D1 dan komputer Ubuntu Linux.

Catatan penting:

Gunakan kabel USB data, bukan kabel pengisi daya saja. Jika board menyala tetapi port serial tidak muncul, kemungkinan kabel tidak mendukung data.

Lampiran B — Gambar Sambungan WeMOS mini D1

Lampiran ini menjadi panduan cepat sambungan dasar. Semua rangkaian memakai prinsip **tegangan benar, GND bersama, dan pin sesuai kode**.

B.1 WeMOS mini D1 ke LED

D5 → Resistor 220Ω / 330Ω → kaki panjang LED
kaki pendek LED → GND

Fungsi:

LED menjadi output digital. Saat D5 bernilai HIGH, LED menyala. Saat D5 LOW, LED mati.

Catatan:

LED wajib memakai resistor agar arus tidak berlebihan.

B.2 WeMOS mini D1 ke Button

D6 → Push Button → GND
Mode program: INPUT_PULLUP

Fungsi:

Button menjadi input digital. Dengan **INPUT_PULLUP**, saat tombol ditekan biasanya terbaca **LOW**.

Logika:

Kondisi	Nilai terbaca
Tombol dilepas	HIGH
Tombol ditekan	LOW

B.3 WeMOS mini D1 ke Potensiometer

3.3V → kaki luar potensiometer
A0 → kaki tengah potensiometer
GND → kaki luar potensiometer

Fungsi:

Potensiometer menjadi input analog. Saat diputar, tegangan di kaki tengah berubah dan dibaca oleh A0.

Catatan keras:

A0 hanya untuk input analog. Jangan masukkan tegangan melebihi batas board.

B.4 WeMOS mini D1 ke Relay DC

D7 → IN relay
3.3V / 5V sesuai modul → VCC relay
GND → GND relay

Fungsi:

Relay menjadi saklar elektronik untuk beban DC kecil.

Catatan keselamatan:

Untuk siswa pemula, gunakan **beban DC rendah**. Jangan menggunakan beban berbahaya tanpa pengawasan guru dan prosedur keselamatan.

B.5 WeMOS mini D1 ke Serial Komputer

WeMOS mini D1 → Kabel USB data → Komputer Ubuntu Linux →
Arduino IDE → Serial Monitor

Fungsi:

Serial dipakai untuk upload program, melihat pesan debug, membaca data waktu, dan mengirim perintah seperti **START**, **STOP**, **RESET**.

Lampiran C — Pinout WeMOS mini D1

WeMOS mini D1 memiliki pin digital, analog, serial, daya, dan ground. Siswa wajib memahami pin sebelum merangkai.

Pin Board	Fungsi Umum	Catatan
D0	GPIO digital	Sering dikaitkan dengan WAKE / GPIO16.
D1	GPIO digital / SCL	Umum dipakai untuk I2C SCL.
D2	GPIO digital / SDA	Umum dipakai untuk I2C SDA.
D3	GPIO digital	Perlu hati-hati saat <i>boot</i> .
D4	GPIO digital / LED internal pada banyak board	Perlu hati-hati saat <i>boot</i> .
D5	GPIO digital	Cocok untuk LED eksternal.
D6	GPIO digital	Cocok untuk button.
D7	GPIO digital	Cocok untuk relay atau buzzer.
D8	GPIO digital	Perlu hati-hati saat <i>boot</i> .
A0	Input analog	Untuk potensiometer atau sensor analog.
TX	Serial transmit	Mengirim data serial.
RX	Serial receive	Menerima data serial.
3V3	Tegangan 3.3V	Untuk sensor 3.3V.
5V	Jalur daya 5V	Bukan berarti GPIO aman menerima sinyal 5V.
GND	Ground	Semua perangkat harus berbagi ground.

Pin yang Perlu Hati-Hati Saat Boot

Beberapa pin ESP8266 mempengaruhi proses *boot*. Untuk pemula, hindari memakai pin sensitif untuk beban yang dapat menarik sinyal ke kondisi salah saat board menyala.

Rekomendasi praktik awal:

Kebutuhan	Pin yang Disarankan
LED	D5
Button	D6
Relay DC	D7
I2C SDA	D2
I2C SCL	D1
Analog	A0

Golden rule pinout:

Jangan hanya mengikuti warna kabel. Ikuti nama pin, fungsi pin, dan batas tegangannya.

Lampiran D — Perintah Ubuntu Server dan Mosquitto

Lampiran ini berisi perintah dasar untuk server IoT lokal.

D.1 Update Sistem

```
sudo apt update
sudo apt upgrade
```

Penjelasan singkat:

- `sudo` menjalankan perintah dengan hak admin.
- `apt update` memperbarui daftar paket.
- `apt upgrade` memperbarui paket yang sudah terpasang.

D.2 Instalasi Mosquitto

```
sudo apt install mosquitto mosquitto-clients
```

Penjelasan singkat:

- `mosquitto` adalah MQTT Broker.
- `mosquitto-clients` berisi alat uji `mosquitto_pub` dan `mosquitto_sub`.

D.3 Cek Service Mosquitto

```
systemctl status mosquitto
```

Menjalankan Mosquitto:

```
sudo systemctl start mosquitto
```

Mengaktifkan Mosquitto saat server menyala:

```
sudo systemctl enable mosquitto
```

Restart Mosquitto:

```
sudo systemctl restart mosquitto
```

D.4 Subscribe Topic

```
mosquitto_sub -h localhost -t smk/lab1/test
```

Arti perintah:

- `mosquitto_sub` berarti menjadi subscriber.
- `-h localhost` berarti broker berada di komputer yang sama.
- `-t smk/lab1/test` berarti membaca topic tersebut.

D.5 Publish Pesan

```
mosquitto_pub -h localhost -t smk/lab1/test -m "Halo MQTT"
```

Arti perintah:

- `mosquitto_pub` berarti menjadi publisher.
- `-t` menentukan topic.
- `-m` menentukan isi pesan atau *payload*.

D.6 MQTT dengan Username dan Password

Subscribe:

```
mosquitto_sub -h 192.168.1.100 -t smk/lab1/test -u smk_iot -P password_anda
```

Publish:

```
mosquitto_pub -h 192.168.1.100 -t smk/lab1/test -m "Tes aman" -u smk_iot -P password_anda
```

D.7 Firewall Sederhana

Cek status firewall:

```
sudo ufw status
```

Izinkan MQTT:

```
sudo ufw allow 1883/tcp
```

Izinkan SSH:

```
sudo ufw allow ssh
```

Aktifkan firewall:

```
sudo ufw enable
```

Catatan keamanan:

Jangan membuka port MQTT ke internet. Untuk lab SMK, cukup gunakan jaringan lokal.

Lampiran E — Kode Lengkap

Lampiran ini berisi kode dasar untuk praktik **WeMOS mini D1 / LOLIN D1 mini**. Urutan kode dibuat bertahap: mulai dari LED, button, potensiometer, serial clock, Wi-Fi, MQTT, sampai proyek akhir.

Catatan penting sebelum memakai kode:

- Ganti `Lab-IoT-SMK` dengan nama Wi-Fi lab.
- Ganti `password_wifi_lab` dengan password Wi-Fi.
- Ganti `192.168.1.100` dengan IP Ubuntu Server.
- Ganti `smk_iot` dan `password_anda` sesuai akun MQTT.
- Gunakan kabel USB data.
- Pastikan board dipilih: **LOLIN(WEMOS) D1 R2 & mini**.

E.1 Blink LED Internal

Kode ini menyalakan dan mematikan LED bawaan board.

```
void setup() {  
    pinMode(LED_BUILTIN, OUTPUT);  
}  
  
void loop() {  
    digitalWrite(LED_BUILTIN, LOW);  
    delay(1000);  
  
    digitalWrite(LED_BUILTIN, HIGH);  
    delay(1000);  
}
```

Penjelasan singkat:

- `setup()` berjalan sekali saat board menyala.
- `loop()` berjalan berulang terus-menerus.
- `pinMode()` mengatur pin sebagai output.
- `digitalWrite()` mengubah kondisi pin.
- `delay(1000)` memberi jeda 1 detik.

Catatan:

Pada banyak board ESP8266, LED internal menyala saat diberi nilai `LOW`.

E.2 Blink LED Eksternal

Kode ini menyalakan LED eksternal pada pin **D5**.

Sambungan:

D5 → resistor 220Ω / 330Ω → kaki panjang LED
kaki pendek LED → GND

Kode:

```
int ledPin = D5;

void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
  digitalWrite(ledPin, HIGH);
  delay(1000);

  digitalWrite(ledPin, LOW);
  delay(1000);
}
```

Catatan penting:

LED harus memakai resistor. Jangan hubungkan LED langsung ke pin tanpa resistor.

E.3 Button

Kode ini membaca push button pada pin **D6**.

Sambungan:

D6 → push button → GND

Kode memakai mode **INPUT_PULLUP**, sehingga:

Kondisi	Nilai Terbaca
Tombol dilepas	HIGH
Tombol ditekan	LOW

Kode:

```
int buttonPin = D6;

void setup() {
  Serial.begin(9600);
  pinMode(buttonPin, INPUT_PULLUP);
}

void loop() {
  int statusTombol = digitalRead(buttonPin);

  if (statusTombol == LOW) {
    Serial.println("Tombol ditekan");
  } else {
    Serial.println("Tombol tidak ditekan");
  }

  delay(300);
}
```

Penjelasan singkat:

- **Serial.begin(9600)** membuka komunikasi serial.
- **digitalRead()** membaca status pin.
- **INPUT_PULLUP** memakai resistor *pull-up* internal.
- **Serial.println()** menampilkan teks ke Serial Monitor.

E.4 Potensiometer

Kode ini membaca nilai analog dari potensiometer melalui pin **A0**.

Sambungan:

3.3V → kaki luar potensiometer
A0 → kaki tengah potensiometer
GND → kaki luar potensiometer

Kode:

```
void setup() {  
    Serial.begin(9600);  
}  
  
void loop() {  
    int nilaiPot = analogRead(A0);  
  
    Serial.print("Nilai potensiometer: ");  
    Serial.println(nilaiPot);  
  
    delay(300);  
}
```

Catatan penting:

Gunakan 3.3V untuk praktik ini. Jangan memberi tegangan berlebih ke A0.

E.5 Serial Clock

Kode ini membuat jam digital sederhana berbasis `millis()`.

Jam ini bukan jam waktu nyata. Jam akan mulai dari nol saat board dinyalakan atau di-reset.

Kode:

```
unsigned long waktuAwal = 0;  
  
void setup() {  
    Serial.begin(9600);  
    waktuAwal = millis();  
  
    Serial.println("Jam digital sederhana dimulai");  
}
```

```

void loop() {
    unsigned long totalDetik = (millis() - waktuAwal) / 1000;

    int detik = totalDetik % 60;
    int menit = (totalDetik / 60) % 60;
    int jam = (totalDetik / 3600) % 24;

    if (jam < 10) Serial.print("0");
    Serial.print(jam);
    Serial.print(":");

    if (menit < 10) Serial.print("0");
    Serial.print(menit);
    Serial.print(":");

    if (detik < 10) Serial.print("0");
    Serial.println(detik);

    delay(1000);
}

```

Output contoh di Serial Monitor:

```

00:00:01
00:00:02
00:00:03

```

Penjelasan singkat:

- `millis()` menghitung waktu sejak program mulai.
- `/ 1000` mengubah milidetik menjadi detik.
- `% 60` mengambil sisa detik atau menit.
- Format `HH:MM:SS` dibuat dengan `Serial.print()`.

E.6 Wi-Fi Connect

Kode ini menghubungkan WeMOS mini D1 ke jaringan Wi-Fi.

```
#include <ESP8266WiFi.h>

const char* ssid = "Lab-IoT-SMK";
const char* password = "password_wifi_lab";

void setup() {
  Serial.begin(9600);
  delay(1000);

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  Serial.print("Menghubungkan ke Wi-Fi");

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println();
  Serial.println("Wi-Fi tersambung");
  Serial.print("IP WeMOS mini D1: ");
  Serial.println(WiFi.localIP());
}

void loop() {
}
```

Penjelasan singkat:

- `#include <ESP8266WiFi.h>` memanggil library Wi-Fi ESP8266.
- `ssid` adalah nama jaringan Wi-Fi.
- `password` adalah password Wi-Fi.
- `WIFI_STA` membuat WeMOS menjadi klien Wi-Fi.
- `WiFi.localIP()` menampilkan IP WeMOS.

E.7 MQTT Publish

Kode ini mengirim nilai potensiometer ke MQTT Broker.

Topic:

smk/lab1/potensio

Kode:

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

const char* ssid = "Lab-IoT-SMK";
const char* password = "password_wifi_lab";

const char* mqtt_server = "192.168.1.100";
const char* mqtt_user = "smk_iot";
const char* mqtt_pass = "password_anda";

WiFiClient espClient;
PubSubClient client(espClient);

void setup_wifi() {
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  Serial.print("Menghubungkan Wi-Fi");

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println();
  Serial.print("IP WeMOS: ");
  Serial.println(WiFi.localIP());
}

void reconnect() {
  while (!client.connected()) {
    Serial.print("Menghubungkan MQTT...");

    if (client.connect("wemos_publish_lab1", mqtt_user,
mqtt_pass)) {
      Serial.println("terhubung");
    } else {
      Serial.print("gagal, rc=");
    }
  }
}
```

```

        Serial.println(client.state());
        delay(5000);
    }
}

void setup() {
    Serial.begin(9600);
    setup_wifi();

    client.setServer(mqtt_server, 1883);
}

void loop() {
    if (!client.connected()) {
        reconnect();
    }

    client.loop();

    int nilaiPot = analogRead(A0);

    char payload[10];
    sprintf(payload, "%d", nilaiPot);

    client.publish("smk/lab1/potensio", payload);

    Serial.print("Publish potensio: ");
    Serial.println(payload);

    delay(1000);
}

```

Penjelasan singkat:

- `PubSubClient` dipakai untuk MQTT.
- `client.setServer()` menentukan alamat broker.
- `client.connect()` menghubungkan WeMOS ke broker.
- `client.publish()` mengirim data ke topic.
- `sprintf()` mengubah angka menjadi teks.

E.8 MQTT Subscribe

Kode ini menerima perintah dari Android untuk menyalakan atau mematikan LED.

Topic:

smk/lab1/led/set

Payload:

ON
OFF

Kode:

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

const char* ssid = "Lab-IoT-SMK";
const char* password = "password_wifi_lab";

const char* mqtt_server = "192.168.1.100";
const char* mqtt_user = "smk_iot";
const char* mqtt_pass = "password_anda";

int ledPin = D5;

WiFiClient espClient;
PubSubClient client(espClient);

void callback(char* topic, byte* payload, unsigned int
length) {
    String pesan = "";

    for (unsigned int i = 0; i < length; i++) {
        pesan += (char)payload[i];
    }

    Serial.print("Topic: ");
    Serial.println(topic);

    Serial.print("Pesan: ");
    Serial.println(pesan);

    if (pesan == "ON") {
        digitalWrite(ledPin, HIGH);
        client.publish("smk/lab1/led/status", "ON");
    }
}
```

```

    if (pesan == "OFF") {
        digitalWrite(ledPin, LOW);
        client.publish("smk/lab1/led/status", "OFF");
    }
}

void setup_wifi() {
    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, password);

    Serial.print("Menghubungkan Wi-Fi");

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }

    Serial.println();
    Serial.print("IP WeMOS: ");
    Serial.println(WiFi.localIP());
}

void reconnect() {
    while (!client.connected()) {
        Serial.print("Menghubungkan MQTT...");

        if (client.connect("wemos_subscribe_lab1", mqtt_user,
mqtt_pass)) {
            Serial.println("terhubung");
            client.subscribe("smk/lab1/led/set");
        } else {
            Serial.print("gagal, rc=");
            Serial.println(client.state());
            delay(5000);
        }
    }
}

void setup() {
    Serial.begin(9600);

    pinMode(ledPin, OUTPUT);
    digitalWrite(ledPin, LOW);

    setup_wifi();

    client.setServer(mqtt_server, 1883);
    client.setCallback(callback);
}

```

```
}

void loop() {
  if (!client.connected()) {
    reconnect();
  }

  client.loop();
}
```

Penjelasan singkat:

- `client.subscribe()` membaca topic MQTT.
- `callback()` berjalan saat pesan masuk.
- Payload `ON` menyalakan LED.
- Payload `OFF` mematikan LED.
- Status LED dikirim ulang ke `smk/lab1/led/status`.

E.9 Proyek Akhir — Smart IoT Learning Panel

Kode ini menggabungkan:

- LED dikontrol dari Android.
- Button mengirim status ke MQTT.
- Potensiometer mengirim nilai analog.
- Jam digital tampil di Serial Monitor.
- Jam digital dikirim ke MQTT.
- Android menampilkan semua data.

Sambungan

LED:

D5 → resistor 220Ω / 330Ω → LED → GND

Button:

D6 → push button → GND

Potensiometer:

3.3V → kaki luar potensiometer

A0 → kaki tengah potensiometer

GND → kaki luar potensiometer

Topic MQTT

Fungsi	Topic	Arah
LED command	<code>smk/lab1/led/set</code>	Android → WeMOS
LED status	<code>smk/lab1/led/status</code>	WeMOS → Android
Button	<code>smk/lab1/button</code>	WeMOS → Android
Potensiometer	<code>smk/lab1/potensio</code>	WeMOS → Android
Jam digital	<code>smk/lab1/jam</code>	WeMOS → Android
Status perangkat	<code>smk/lab1/status</code>	WeMOS → Android

Kode:

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

const char* ssid = "Lab-IoT-SMK";
const char* password = "password_wifi_lab";

const char* mqtt_server = "192.168.1.100";
const char* mqtt_user = "smk_iot";
```

```

const char* mqtt_pass = "password_anda";

int ledPin = D5;
int buttonPin = D6;

int statusButtonLama = HIGH;

unsigned long waktuAwal = 0;
unsigned long waktuKirim = 0;

WiFiClient espClient;
PubSubClient client(espClient);

void setup_wifi() {
    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, password);

    Serial.print("Menghubungkan Wi-Fi");

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }

    Serial.println();
    Serial.print("IP WeMOS: ");
    Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* payload, unsigned int
length) {
    String pesan = "";

    for (unsigned int i = 0; i < length; i++) {
        pesan += (char)payload[i];
    }

    Serial.print("Pesan MQTT masuk: ");
    Serial.println(pesan);

    if (pesan == "ON") {
        digitalWrite(ledPin, HIGH);
        client.publish("smk/lab1/led/status", "ON");
    }

    if (pesan == "OFF") {
        digitalWrite(ledPin, LOW);
        client.publish("smk/lab1/led/status", "OFF");
    }
}

```

```

    }
}

void reconnect() {
    while (!client.connected()) {
        Serial.print("Menghubungkan MQTT...");

        if (client.connect("smart_panel_lab1", mqtt_user,
mqtt_pass)) {
            Serial.println("terhubung");

            client.subscribe("smk/lab1/led/set");
            client.publish("smk/lab1/status", "ONLINE");
        } else {
            Serial.print("gagal, rc=");
            Serial.println(client.state());
            delay(5000);
        }
    }
}

void kirimButton() {
    int statusButtonBaru = digitalRead(buttonPin);

    if (statusButtonBaru != statusButtonLama) {
        if (statusButtonBaru == LOW) {
            client.publish("smk/lab1/button", "PRESSED");
            Serial.println("Button: PRESSED");
        } else {
            client.publish("smk/lab1/button", "RELEASED");
            Serial.println("Button: RELEASED");
        }

        statusButtonLama = statusButtonBaru;
        delay(200);
    }
}

void kirimPotensio() {
    int nilaiPot = analogRead(A0);

    char payload[10];
    sprintf(payload, "%d", nilaiPot);

    client.publish("smk/lab1/potensio", payload);

    Serial.print("Potensio: ");
    Serial.println(payload);
}

```

```

}

void kirimJam() {
    unsigned long totalDetik = (millis() - waktuAwal) / 1000;

    int detik = totalDetik % 60;
    int menit = (totalDetik / 60) % 60;
    int jam = (totalDetik / 3600) % 24;

    char waktu[10];
    sprintf(waktu, "%02d:%02d:%02d", jam, menit, detik);

    client.publish("smk/lab1/jam", waktu);

    Serial.print("Jam: ");
    Serial.println(waktu);
}

void setup() {
    Serial.begin(9600);

    pinMode(ledPin, OUTPUT);
    pinMode(buttonPin, INPUT_PULLUP);

    digitalWrite(ledPin, LOW);

    waktuAwal = millis();

    setup_wifi();

    client.setServer(mqtt_server, 1883);
    client.setCallback(callback);
}

void loop() {
    if (!client.connected()) {
        reconnect();
    }

    client.loop();

    kirimButton();

    if (millis() - waktuKirim >= 1000) {
        kirimPotensio();
        kirimJam();
        waktuKirim = millis();
    }
}

```

}

Checklist Uji Proyek Akhir

Fitur	Topic	Hasil yang Diharapkan
LED ON	<code>smk/lab1/led/set</code>	LED menyala
LED OFF	<code>smk/lab1/led/set</code>	LED mati
Status LED	<code>smk/lab1/led/status</code>	Android menampilkan ON/OFF
Button	<code>smk/lab1/button</code>	Android menampilkan PRESSED/RELEASED
Potensiometer	<code>smk/lab1/potensio</code>	Android menampilkan angka
Jam digital	<code>smk/lab1/jam</code>	Android menampilkan <code>HH:MM:SS</code>
Status perangkat	<code>smk/lab1/status</code>	Android menampilkan ONLINE

Penutup Lampiran E:

Kode bukan untuk dihafal, tetapi untuk dipahami, diuji, dimodifikasi, dan dijadikan dasar proyek IoT SMK yang lebih kreatif.

Lampiran F — Checklist Troubleshooting

F.1 Upload Program

- Board sudah dipilih **LOLIN(WEMOS) D1 R2 & mini**.
- Port serial sudah benar.
- Kabel USB adalah kabel data.
- Rangkaian eksternal dilepas jika upload gagal.
- Program tidak ada error kompilasi.
- Tekan reset jika diperlukan.

F.2 Serial Port

- Cek port:

```
ls /dev/ttyUSB*  
ls /dev/ttyACM*
```

- Cabut-pasang kabel USB.
- Coba port USB lain.
- Coba kabel data lain.
- Pastikan user memiliki izin akses serial.

F.3 Wi-Fi

- SSID benar.
- Password benar.
- Router menyala.
- Jarak board dekat dengan router.
- Gunakan jaringan 2.4 GHz.
- Serial Monitor menampilkan IP address.
- Cek kekuatan sinyal.

F.4 MQTT

- Mosquitto aktif.
- IP broker benar.
- Port 1883 terbuka.
- Username benar.
- Password benar.
- Topic sama persis.
- Payload sesuai kode.
- Uji dengan `mosquitto_pub` dan `mosquitto_sub`.

F.5 Android Dashboard

- Android satu Wi-Fi dengan broker.
- IP broker benar.
- Port benar.
- Username/password benar.
- Widget subscribe memakai topic data.
- Widget publish memakai topic perintah.
- Payload **ON** dan **OFF** ditulis sesuai kode.

F.6 Sensor

- Pin sesuai kode.
- GND tersambung.
- Tegangan sesuai.
- Sensor tidak terbalik.
- A0 tidak menerima tegangan berlebih.
- Serial Monitor menampilkan nilai mentah.

F.7 Power

- Kabel USB stabil.
- Board tidak restart sendiri.
- Beban tidak mengambil arus besar dari pin.
- Relay atau servo memakai daya sesuai kebutuhan.
- GND disatukan bila memakai catu daya eksternal.

F.8 Ground

- Semua perangkat berbagi GND.
- WeMOS GND terhubung ke GND sensor.
- WeMOS GND terhubung ke GND relay jika relay dipakai.
- GND power eksternal disatukan dengan GND WeMOS bila ada sinyal bersama.

Prinsip troubleshooting:

Cek dari fisik dulu: daya, kabel, GND, pin. Setelah itu baru cek kode, Wi-Fi, MQTT, dan dashboard.

Lampiran G — Template Laporan Proyek IoT SMK

Judul Proyek

Tuliskan nama proyek secara jelas.

Contoh:

Smart IoT Learning Panel dengan WeMOS mini D1

1. Tujuan

Tuliskan tujuan proyek.

Contoh:

- Membuat LED dapat dikontrol dari Android.
- Mengirim status button ke MQTT.
- Menampilkan nilai potensiometer di dashboard.
- Menampilkan jam digital melalui Serial Monitor dan Android.

2. Komponen

No	Komponen	Jumlah	Fungsi
1	WeMOS mini D1	1	Board utama
2	LED	1	Output digital
3	Resistor 220Ω	1	Pembatas arus LED
4	Push button	1	Input digital
5	Potensiometer	1	Input analog
6	Breadboard	1	Papan rangkaian
7	Kabel jumper	secukupnya	Penghubung
8	Ubuntu Server	1	MQTT Broker
9	Android	1	Dashboard

3. Diagram

Sisipkan:

- Diagram rangkaian.
- Diagram MQTT.
- Diagram topic.

4. Topic MQTT

Fungsi	Topic	Payload
LED command	<code>smk/lab1/led/set</code>	ON / OFF
LED status	<code>smk/lab1/led/status</code>	ON / OFF
Button	<code>smk/lab1/button</code>	PRESSED / RELEASED
Potensiometer	<code>smk/lab1/potensio</code>	angka
Jam digital	<code>smk/lab1/jam</code>	HH:MM:SS

5. Kode

Tempelkan kode program utama. Beri keterangan:

- Pin yang dipakai.
- Topic yang dipakai.
- Fungsi utama program.
- Cara koneksi Wi-Fi.
- Cara koneksi MQTT.

6. Hasil Uji

Pengujian	Hasil	Status
LED ON dari Android	LED menyala	Berhasil / Gagal
LED OFF dari Android	LED mati	Berhasil / Gagal
Button ditekan	Android tampil PRESSED	Berhasil / Gagal
Potensiometer diputar	Nilai berubah	Berhasil / Gagal
Jam digital	Format tampil	Berhasil / Gagal
MQTT broker	Publish/subscribe berjalan	Berhasil / Gagal

7. Masalah

Tuliskan masalah yang ditemukan.

Contoh:

- Android tidak menerima data.
- MQTT gagal konek.
- LED tidak menyala.
- Port serial tidak muncul.
- Nilai A0 tidak berubah.

8. Solusi

Tuliskan solusi yang dilakukan.

Contoh:

- Mengganti kabel USB data.
- Memperbaiki topic MQTT.
- Mengecek IP broker.
- Menyamakan username/password.
- Memperbaiki sambungan GND.

9. Kesimpulan

Tuliskan 1–2 paragraf.

Contoh:

Proyek Smart IoT Learning Panel berhasil menunjukkan bahwa WeMOS mini D1 dapat membaca button dan potensiometer, mengontrol LED, mengirim data jam digital, dan berkomunikasi dengan Android melalui MQTT Broker lokal. Proyek ini membuktikan bahwa IoT sederhana dapat dibuat dengan komponen murah, server lokal, dan dokumentasi yang rapi.

Glossary IoT Dasar

Istilah	Arti
A0	Pin input analog pada WeMOS mini D1 untuk membaca tegangan analog, misalnya dari potensiometer.
ADC	<i>Analog-to-Digital Converter</i> , bagian yang mengubah sinyal analog menjadi angka digital.
Android Dashboard	Tampilan di HP Android untuk membaca data sensor dan mengirim perintah kontrol melalui MQTT.
Arduino IDE	Aplikasi open source untuk menulis, mengunggah, dan menguji program pada board seperti WeMOS mini D1.
Baud Rate	Kecepatan komunikasi serial, misalnya 9600 atau 115200 bps.
Board	Papan mikrokontroler yang berisi chip, pin, regulator daya, dan konektor USB.
Broker	Pusat pesan dalam MQTT yang menerima pesan dari <i>publisher</i> dan meneruskannya ke <i>subscriber</i> .
Breadboard	Papan percobaan tanpa solder untuk merangkai komponen elektronik sementara.
Buzzer	Komponen output suara untuk alarm, indikator, atau notifikasi sederhana.
Client ID	Nama identitas perangkat atau aplikasi saat terhubung ke MQTT Broker.
Dashboard	Tampilan visual untuk monitoring data dan kontrol perangkat IoT.
Daya	Sumber energi listrik yang membuat board dan komponen dapat bekerja.
ESP8266	Chip Wi-Fi yang menjadi dasar WeMOS mini D1 dan cocok untuk proyek IoT sederhana.
ESP8266 Core	Paket tambahan di Arduino IDE agar board ESP8266 seperti WeMOS mini D1 dapat diprogram.
ESPHome	Tool open source berbasis konfigurasi YAML untuk membuat perangkat IoT berbasis ESP8266/ESP32.
Firewall	Sistem pengaman jaringan untuk mengatur port dan koneksi yang boleh masuk ke server.

GPIO	<i>General Purpose Input/Output</i> , pin digital yang dapat dipakai sebagai input atau output.
GND	<i>Ground</i> , titik referensi nol volt dalam rangkaian elektronik.
Ground Bersama	Kondisi ketika semua perangkat yang saling bertukar sinyal memiliki jalur GND yang sama.
HIGH	Kondisi logika digital aktif atau bernilai tinggi.
InfluxDB	Database <i>time-series</i> open source untuk menyimpan data sensor berdasarkan waktu.
Input	Data atau sinyal yang masuk ke board, misalnya button atau potensiometer.
Input Analog	Input dengan nilai berubah bertahap, misalnya tegangan dari potensiometer.
Input Digital	Input dengan dua keadaan, misalnya ditekan/tidak ditekan atau HIGH/LOW.
IoT	<i>Internet of Things</i> , konsep menghubungkan benda fisik ke jaringan agar dapat membaca data, mengirim data, dan menerima perintah.
IP Address	Alamat perangkat dalam jaringan, misalnya 192.168.1.45.
Jam Digital	Tampilan waktu dalam format angka, misalnya 00:05:21.
Jaringan Lokal	Jaringan internal di sekolah, rumah, atau lab yang menghubungkan perangkat tanpa harus bergantung ke internet luar.
Kabel Jumper	Kabel kecil untuk menghubungkan pin board dengan breadboard atau komponen lain.
Kabel USB Data	Kabel USB yang dapat dipakai untuk upload program dan komunikasi serial, bukan hanya mengisi daya.
LED	<i>Light Emitting Diode</i> , komponen output cahaya untuk indikator sederhana.
LOLIN D1 mini	Nama lain / varian resmi board kecil berbasis ESP8266 yang sering disebut WeMOS mini D1.
LOW	Kondisi logika digital tidak aktif atau bernilai rendah.
MQTT	Protokol pesan ringan berbasis <i>publish/subscribe</i> yang banyak dipakai dalam IoT.
Mosquitto	MQTT Broker open source yang dapat dijalankan di Ubuntu Server.

Node-RED	Tool open source berbasis alur visual untuk menghubungkan MQTT, sensor, database, dan dashboard.
NTP	<i>Network Time Protocol</i> , protokol untuk mengambil waktu dari server jaringan.
Output	Sinyal atau aksi yang keluar dari board, misalnya LED menyala atau relay aktif.
Output Digital	Output dengan dua keadaan, misalnya ON/OFF atau HIGH/LOW.
Payload	Isi pesan MQTT, misalnya ON , OFF , 735 , atau 00:05:21 .
Pin	Kaki koneksi pada board untuk daya, ground, input, output, atau komunikasi data.
Pinout	Peta posisi dan fungsi pin pada board.
Port 1883	Port default MQTT tanpa TLS yang umum dipakai untuk komunikasi MQTT lokal.
Potensiometer	Komponen input analog yang menghasilkan perubahan tegangan saat diputar.
Publisher	Perangkat atau aplikasi yang mengirim pesan ke topic MQTT.
Pull-down	Teknik membuat input default LOW menggunakan resistor ke GND.
Pull-up	Teknik membuat input default HIGH menggunakan resistor ke tegangan positif.
PWM	<i>Pulse Width Modulation</i> , teknik mengatur output digital agar terlihat seperti analog, misalnya meredupkan LED.
Relay Module DC	Modul saklar elektronik untuk mengontrol beban DC kecil.
Resistor	Komponen pembatas arus, penting untuk melindungi LED dan pin board.
RTC	<i>Real-Time Clock</i> , modul jam fisik yang dapat menjaga waktu walau board mati.
Serial Monitor	Jendela di Arduino IDE untuk melihat pesan dari board dan mengirim perintah ke board.
Serial Port	Jalur komunikasi data antara board dan komputer, biasanya melalui USB.
Server Lokal	Komputer di jaringan sekolah/lab yang menjalankan layanan seperti Mosquitto MQTT Broker.

SSID	Nama jaringan Wi-Fi.
Subscriber	Perangkat atau aplikasi yang membaca pesan dari topic MQTT.
Topic	Alamat pesan dalam MQTT, misalnya <code>smk/lab1/potensio</code> .
TX/RX	Pin komunikasi serial: TX untuk mengirim data, RX untuk menerima data.
UART	Protokol komunikasi serial yang memakai jalur TX dan RX.
Ubuntu Server	Sistem operasi server open source yang dipakai untuk menjalankan layanan IoT lokal seperti Mosquitto.
WeMOS mini D1	Board kecil berbasis ESP8266 dengan Wi-Fi bawaan untuk belajar IoT sederhana.
Wi-Fi	Jaringan nirkabel yang menghubungkan WeMOS mini D1, router, server, laptop, dan Android.
Wi-Fi Station Mode	Mode ketika WeMOS mini D1 tersambung ke router seperti laptop atau HP.
Widget	Komponen tampilan pada dashboard Android, misalnya tombol ON/OFF, indikator status, atau tampilan angka.

Referensi

Arduino. (2024, February 12). *Using the Serial Monitor tool*. Arduino Documentation. <https://docs.arduino.cc/software/ide-v2/tutorials/ide-v2-serial-monitor> (Dokumentasi Arduino)

Arduino. (2024, November 20). *Serial*. Arduino Documentation. <https://docs.arduino.cc/language-reference/en/functions/communication/serial/> (Dokumentasi Arduino)

Arduino. (2025, July 30). *Downloading and installing the Arduino IDE 2*. Arduino Documentation. <https://docs.arduino.cc/software/ide-v2/tutorials/getting-started/ide-v2-downloading-and-installing> (Dokumentasi Arduino)

Arduino. (n.d.). *PubSubClient*. Arduino Documentation. <https://docs.arduino.cc/libraries/pubsubclient/> (Dokumentasi Arduino)

Arduino. (n.d.). *Getting started with Arduino IDE 2*. Arduino Documentation. <https://docs.arduino.cc/software/ide-v2/tutorials/getting-started-ide-v2> (Dokumentasi Arduino)

Eclipse Foundation. (n.d.). *Eclipse Mosquitto: An open source MQTT broker*. Eclipse Mosquitto. <https://mosquitto.org/> (Eclipse Mosquitto)

Eclipse Foundation. (n.d.). *Eclipse Mosquitto documentation*. Eclipse Mosquitto. <https://mosquitto.org/documentation/> (Eclipse Mosquitto)

Eclipse Foundation. (n.d.). *Authentication methods*. Eclipse Mosquitto. <https://mosquitto.org/documentation/authentication-methods/> (Eclipse Mosquitto)

Eclipse Foundation. (n.d.). *mosquitto.conf man page*. Eclipse Mosquitto. <https://mosquitto.org/man/mosquitto-conf-5.html> (Eclipse Mosquitto)

ESP8266 Arduino Core. (n.d.). *ESP8266 Arduino Core documentation*. Read the Docs. <https://arduino-esp8266.readthedocs.io/> (arduino-esp8266.readthedocs.io)

ESP8266 Arduino Core. (n.d.). *Installing with Boards Manager*. Read the Docs. <https://arduino-esp8266.readthedocs.io/en/latest/installing.html> (arduino-esp8266.readthedocs.io)

ESP8266 Arduino Core. (n.d.). *ESP8266WiFi library*. Read the Docs. <https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html> (cookbook.nodered.org)

ESPHome. (n.d.). *ESPHome: Smart home made simple*. <https://esphome.io/> (ESPHome - Smart Home Made Simple)

Grafana Labs. (n.d.). *Dashboards*. Grafana Documentation. <https://grafana.com/docs/grafana/latest/visualizations/dashboards/> (Grafana Labs)

InfluxData. (n.d.). *InfluxDB*. <https://www.influxdata.com/> ([influxdata.com](https://www.influxdata.com/))

Knolleary. (n.d.). *Arduino Client for MQTT*. PubSubClient. <https://pubsubclient.knolleary.net/> (pubsubclient.knolleary.net)

LOLIN. (n.d.). *LOLIN D1 mini*. WEMOS Documentation. https://www.wemos.cc/en/latest/d1/d1_mini.html ([wemos.cc](https://www.wemos.cc/))

LOLIN. (n.d.). *D1 mini shields*. WEMOS Documentation. https://www.wemos.cc/en/latest/d1_mini_shield/index.html ([wemos.cc](https://www.wemos.cc/))

MQTT.org. (n.d.). *MQTT specification*. <https://mqtt.org/mqtt-specification/> ([MQTT](https://mqtt.org/))

Node-RED. (n.d.). *Connect to an MQTT broker*. Node-RED Cookbook. <https://cookbook.nodered.org/mqtt/connect-to-broker> (cookbook.nodered.org)

OASIS Open. (2019). *MQTT Version 5.0: OASIS Standard*. <https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html> (docs.oasis-open.org)

OASIS Open. (2019). *MQTT Version 5.0*. <https://www.oasis-open.org/standard/mqtt-v5-0-os/> ([OASIS Open](https://www.oasis-open.org/))

Purbo, O. W. (2026). *IoT Sederhana dengan WeMOS mini D1 untuk SMK*. Naskah buku tidak diterbitkan.

Ubuntu. (2025, December 11). *Firewall*. Ubuntu Server Documentation. <https://ubuntu.com/server/docs/how-to/security/firewalls/> ([Ubuntu](https://ubuntu.com/))

Ubuntu. (2026, March 23). *OpenSSH server*. Ubuntu Server Documentation. <https://ubuntu.com/server/docs/how-to/security/openssh-server/> ([Ubuntu](https://ubuntu.com/))

Ubuntu. (2026, March 26). *Configuring networks*. Ubuntu Server Documentation. <https://ubuntu.com/server/docs/explanation/networking/configuring-networks/> ([Ubuntu](https://ubuntu.com/))

Tentang Penulis



Onno W. Purbo, saat ini bertugas sebagai rektor di Institut Teknologi Tangerang Selatan (ITTS). Onno memperoleh gelar Ph.D bidang Electrical Engineering dari University of Waterloo, Canada, adalah seorang copyleft, educator dan ICT evangelist. Dia sudah mempublikasikan 50++ buku, termasuk free ICT ebook untuk sekolah tahun 2008. Beberapa buku terakhirnya adalah "Internet-TCP/IP: Konsep Dan Implementasi", 2018; "Sistem Operasi, Konsep Dan Membuat Linux OpenWRT Dan ROM Android", 2019; "IPv6 Untuk Mendukung Operasi Jaringan Dan Domain Name System", 2019; "Kubernetes untuk Pemula", 2024 dan "Membuat Operator Seluler 5G Sendiri", 2024.

Dia memimpin sambungan pertama Internet di Institut Teknologi Bandung, tahun 1993-2000, dan menggunakannya untuk membuat jaringan Internet pendidikan yang pertama di Indonesia. Dia membebaskan frekuensi WiFi, memperkenalkan RT/RW-net, antenna Wajanbolic dan jaringan selular OpenBTS dan private 5G sendiri. Dia memimpin jaringan telepon pertama di atas Internet, VoIP Merdeka, yang kemudian hari dikenal sebagai VoIP Rakyat berbasis SIP dan menggunakan kode area +62520 dan +62521. Dia saat ini aktif memperkenalkan e-Learning, dan menjalankan server e-Learning di <http://lms.onnocenter.or.id/moodle/> 66,000++ siswa / mahasiswa dan <https://opencourse.itts.ac.id> 39.000+ mahasiswa secara gratis. Email: onno@indo.net.id