BAB II

TINJAUAN PUSTAKA

2.1 Justifikasi Solusi

Menindaklanjuti permasalahan yang telah diidentifikasi pada Bab I, perancangan ulang *IoT node* MySalak V2 memerlukan landasan ilmiah yang kuat dan pendekatan rekayasa yang metodologis. Keberhasilan dalam mencapai target efisiensi daya bergantung pada dua fondasi utama yaitu, penguasaan konsep-konsep teoritis yang relevan dan analisis kritis terhadap solusi-solusi praktis yang telah ada dan analisis *post-mortem* terhadap kegagalan daya pada *node* V1 berfungsi sebagai kerangka acuan praktis untuk mengevaluasi dan memilih teknologi yang paling sesuai dari berbagai alternatif yang ditinjau.

2.1.1 Desain Perangkat IoT Mandiri Jarak Jauh (Kjellby et al., 2018) [2]

Dalam penelitian berjudul *"Design and Prototype Implementation of Long-Range Self-Powered Wireless IoT Devices"* oleh Kjellby et al. (2018) mendemonstrasikan perancangan dan implementasi sebuah *node* sensor *wireless* yang sepenuhnya mandiri. Penelitian ini menggunakan *System on Chip* (SoC) nRF52840 yang dikenal sangat hemat daya. Sistem *power supply* mereka dirancang secara cermat menggunakan *Power Management* IC BQ25570 yang mampu memanen daya dari panel surya 0.36W untuk mengisi baterai *lithium-ion coin cell* berkapasitas 120mAh. Melalui optimasi perangkat keras dan perangkat lunak, mereka berhasil mencapai konsumsi daya rata-rata yang sangat rendah, yaitu hanya 10.3µW saat menggunakan interval transmisi 55 detik. Hasil ini menunjukkan bahwa daya yang dipanen 91 kali lebih besar daripada yang dikonsumsi, membuktikan kelayakan operasional jangka panjang hanya dengan baterai berkapasitas sangat kecil.

Justifikasi untuk *IoT Node* v2:

- Sebuah IC *nanopower buck-boost converter* BQ25570 dengan manajemen baterai terintegrasi. Fitur utamanya yang relevan adalah *quiescent current* sangat rendah (900nA) dan *cold start voltage* rendah (330mV). Namun, fitur yang paling krusial dari IC ini adalah fungsionalitas *Maximum Power Point Tracking* (MPPT) yang terintegrasi. Secara aktif, sirkuit MPPT akan terus-menerus menyesuaikan beban pada panel surya untuk memastikan panel tersebut selalu beroperasi pada titik tegangan dan arus di mana ia menghasilkan daya paling maksimal.
- Penelitian ini menjelaskan bahwa interval *duty cycle* terhadap konsumsi daya rata-rata sangat krusial. Menaikkan interval tidur dari beberapa detik ke 55 detik menurunkan konsumsi daya secara eksponensial pada penelitian ini. Karena konsumsi daya paling besar adalah dari transmisi.

2.1.2 Optimisasi Konsumsi Daya untuk Solar-battery IoT berdasarkan Wemos D1 Mini (Gaikwad et al., 2020) [3]

Dalam penelitian berjudul *"Optimizing Power Consumption for Solar Powered Rechargeable Lithium Ion (Li-ion) Battery Operated IoT Based Sensor Node Using WeMos D1 Mini"*, Gaikwad et al. (2020) menyajikan desain dan implementasi *node* IoT bertenaga surya yang fokus pada efisiensi daya. Mikrokontroler yang digunakan adalah WeMos D1 Mini berbasis ESP8266EX yang kompatibel dengan Arduino IDE dan di desain untuk daya rendah. Sistem *power supply* mereka terdiri dari panel surya 6V, baterai Li-Ion 4000mAh, sirkuit pengisian berbasis IC linear TP4056, dan sebuah regulator LDO MCP1700-3302E yang memiliki *quiescent current* sangat rendah. Melalui implementasi mode *deep sleep*, mereka berhasil menekan konsumsi arus hingga sekitar 20µA, dan menyimpulkan bahwa siklus *sleep* selama 10 menit sudah cukup untuk menjaga sistem tetap beroperasi secara berkelanjutan dengan tenaga surya. Mereka

mendapatkan hasil penggunaan daya saat *sleep* hanya 0.627mA/menit dan active time 92.5 mA/menit yang berdurasi hanya 0.5 menit. Sehingga total konsumsi arus di penelitian mereka dalam 10 menit hanya 93.13 mA.

Dengan menggunakan rumus untuk mencari total konsumsi daya dalam 1 cycle,

$$Q_{cycle} = I(mA)_{active} \times t(second)_{active} + I(mA)_{sleep} \times t(second)_{sleep}$$

$$Q_{cycle} = 185mA \times 30s + 0.066mA \times 600s$$

$$Q_{cycle} = 5589.6$$

Lalu karena 1 cycle berlangsung selama T menit, dapat dicari rata-rata konsumsi arus dengan,

$$I_{avg} = \frac{Q_{cycle}}{T(second)}$$

$$I_{avg} = \frac{5589.6}{600}$$

$$I_{avg} = 9.316mA$$

Sehingga dengan menggunakan baterai dalam satuan mAh, dapat dicari masa pakai baterai tersebut dengan,

$$Battery\ Life\ (hours) = \frac{C_{bat}}{I_{avg}}$$

$$Battery\ Life\ (hours) = \frac{4000mAh}{9.316mA}$$

$$Battery\ Life\ (hours) = 429\ hours$$

Estimasi masa pakai tersebut merupakan perhitungan ketahanan baterai dalam skenario statis tanpa adanya pengisian ulang dari panel surya. Dalam kondisi operasional, panel surya yang terpasang mampu menghasilkan arus puncak hingga 250mA di , nilai yang secara signifikan melampaui konsumsi daya rata-rata sistem yang hanya 9.316mA. Hal ini menciptakan *power budget* yang stabil, sehingga memastikan operasional sistem dapat berlangsung secara berkelanjutan selama mendapat paparan cahaya matahari yang memadai.

Justifikasi untuk *IoT Node* v2:

- Gaikwad et al. mengidentifikasi bahwa regulator tegangan biasa boros daya, lalu mereka secara spesifik memilih LDO MCP1700-3302E karena memiliki *quiescent current* yang hanya 1.6µA. Ini adalah kontributor terbesar pada hasil *deep sleep* mereka yang rendah.
- Penggunaan *charger linear* TP4056 adalah kelemahan terbesar dalam desain mereka. TP4056 sangat tidak efisien ketika tegangan input (dari panel surya) jauh lebih tinggi dari tegangan baterai (~3.7V), karena kelebihan energi tersebut dibuang menjadi panas. Sebagai kontras, pendekatan yang jauh lebih superior ditunjukkan oleh Kjellby et al. (2018) melalui penggunaan *Power Management IC* (PMIC) canggih BQ25570 [3]. IC ini mengintegrasikan *switching converter* dengan fungsionalitas Maximum Power Point Tracking (MPPT), yang secara aktif mencari dan beroperasi pada titik tegangan dan arus di mana panel surya menghasilkan daya paling maksimal.

2.1.3 Arsitektur Node Sensor Berbasis Timer Eksternal (Cerchecci et al., 2018) [4]

Dalam penelitian berjudul *"A Low Power IoT Sensor Node Architecture for Waste Management Within Smart Cities Context"*, Cerchecci et al. (2018) menyajikan sebuah arsitektur *IoT node* yang sangat inovatif dengan fokus utama pada efisiensi daya untuk perangkat yang hanya ditenagai baterai. Berbeda dari pendekatan yang mengandalkan mode *deep sleep* internal mikrokontroler, penelitian ini mengimplementasikan kebijakan hemat daya yang ekstrim yaitu dengan menggunakan *timer* eksternal berbasis IC *counter* HEF4060B. *Timer* ini berfungsi sebagai "jam alarm" berdaya sangat rendah (36µA) yang mematikan total seluruh sistem utama, termasuk mikrokontroler ATmega328, selama periode tidak aktif. Ketika *timer* mencapai interval yang ditentukan (sekitar 57 menit), ia akan mengaktifkan sebuah MOSFET yang kemudian menyalakan *switching regulator* Pololu D24V22F5 untuk memberi daya pada mikrokontroler dan sensor. Setelah tugas pengukuran dan transmisi selesai, mikrokontroler akan mengirim sinyal *reset* ke *timer* eksternal, yang secara efektif mematikan dirinya sendiri dan memulai kembali siklus tidur. Dengan arsitektur ini, mereka berhasil mencapai estimasi konsumsi daya rata-rata hanya ~299µA, yang diproyeksikan dapat menghidupi *node* selama lebih dari 500 hari dengan empat baterai AA.

Justifikasi untuk *IoT Node* v2:

- Penelitian ini secara eksplisit membandingkan konsumsi daya desain mereka (berbasis *chip* ATmega328) dengan solusi berbasis *devkit board* Arduino UNO. Hasilnya desain *custom* mereka dua kali lebih tahan lama (502 hari vs. 259 hari) karena pemilihan desain *peripheral* yang lebih baik.

- Sebagai salah satu pilar utama dalam solusi desain keseluruhan, MySalak V2 akan mengadopsi sebuah arsitektur manajemen daya fundamental, yaitu arsitektur *power-off* berbasis *timer* eksternal. Pendekatan ini tidak lagi mengandalkan mode *deep sleep* internal mikrokontroler sebagai kondisi siaga utama, melainkan menggunakan sebuah IC *timer* berdaya sangat rendah yang berfungsi sebagai *watchdog* untuk sistem. Ketika dalam periode tidur, *timer* ini akan memutus total *power supply* ke mikrokontroler dan seluruh *peripheral* utama, sehingga mencapai kondisi daya yang mati total dengan konsumsi arus 0µA untuk bagian sistem yang paling kompleks. Strategi ini terbukti sangat efektif, seperti yang didemonstrasikan oleh Cerchecci et al. (2018), dan secara teoritis menghasilkan kondisi tidur yang lebih dalam dan lebih efisien daripada mode *deep sleep* internal MCU yang masih memiliki kebocoran arus. Dengan demikian, pemilihan arsitektur ini merupakan keputusan desain kunci untuk memastikan MySalak V2 memiliki ketahanan operasional yang semaksimal mungkin.
- Dengan mempertimbangkan aspek *feasibility*, solusi yang diimplementasikan adalah dengan memaksimalkan efisiensi mode *standby* yang dimiliki oleh mikrokontroler STM32U585, yang dikombinasikan dengan kebijakan *power gating* secara menyeluruh dengan timer TPL5111. Pendekatan ini menawarkan peningkatan efisiensi yang sangat signifikan dibandingkan arsitektur V1 dan merupakan solusi paling optimal yang dapat direalisasikan dengan kendala komponen yang ada.

2.1.4 Analisis Post-Mortem *Node* V1

Evaluasi *node* V1 harus dimulai dengan menetapkan kebutuhan operasional esensial di lapangan yaitu ketahanan operasional (*uptime*) jangka panjang. Keterpencilan geografis antara tim *hardware* di Tangerang dan lokasi implementasi di Sleman, Yogyakarta, membuat perawatan rutin menjadi sangat tidak praktis. Oleh karena itu, *node* ditargetkan untuk memiliki *uptime* melampaui 4 bulan agar sinkron dengan jadwal kunjungan. Kebutuhan ini gagal total dipenuhi; data operasional membuktikan seluruh *node* V1 yang telah di-*deploy* mati total , dengan masa pakai rata-rata hanya 60 hingga 70 hari. Performa terbaik (*node* 32) pun hanya bertahan 77.6 hari, sangat jauh dari target.

Untuk mencapai *uptime* tersebut, *node* harus memiliki *power budget* yang berkelanjutan. Arsitektur V1 ditenagai oleh baterai 2x18650 dengan total 7000mAh dan panel surya 1.25W, di mana idealnya daya yang dipanen harus lebih besar dari daya yang dikonsumsi. Namun, mengingat kondisi iklim Sleman yang sering berawan atau hujan, target yang lebih realistis bukanlah *power budget* positif, melainkan *power budget* negatif yang seminimal mungkin. Desain yang efisien seharusnya mampu memperlambat laju penurunan baterai secara drastis. Di sinilah letak kegagalan fundamental V1 yaitu konsumsi dayanya yang sangat besar menciptakan *negative power budget* yang sangat besar, di mana daya yang dipanen tidak mampu mengisi ulang daya yang terkuras oleh konsumsi *standby* yang boros.

Analisis laboratorium mengkonfirmasi besarnya kegagalan ini. Hasil pengujian (dijelaskan di Bab 4) menunjukkan V1 memiliki konsumsi daya rata-rata siklus yang sangat tinggi, yaitu 15.71mA (62.84 mW). Berdasarkan perhitungan teoritis, konsumsi sebesar ini akan menghabiskan baterai 7000mAh hanya dalam 17.17 hari (412.15 jam)

jika beroperasi tanpa panel surya. Perbandingan antara data teoretis (17.17 hari) dan data lapangan (60-77 hari) ini membuktikan bahwa *negative power budget* V1 sangat besar. Panel surya memang berfungsi, namun konsumsi daya V1 begitu boros sehingga daya yang dipanen hanya mampu *memperlambat* laju kematian *node* bukan membuatnya *self-sustain*.

Akar dari *negative power budget* yang besar ini terletak pada serangkaian keputusan desain V1 yang tidak dioptimalkan untuk efisiensi daya. Analisis *post-mortem* mengidentifikasi empat sumber inefisiensi utama, seperti yang dirangkum pada tabel 2.1.

Tabel 2.1 Analisis sumber inefisiensi arsitektur *node* V1

Pendekatan Desain V1	**Analisis Permasalahan**
Arsitektur V1 menggunakan modul DFRobot Solar Power Manager siap pakai, yang berbasis IC *linear charger* CN3065.	*Linear charger* sangat tidak efisien ketika tegangan input jauh lebih tinggi dari tegangan baterai. Kelebihan energi dibuang menjadi panas. Modul ini tidak memiliki *Maximum Power Point Tracking*, sehingga gagal mengekstrak daya maksimal dari panel surya, terutama dalam kondisi cahaya rendah (berawan/hujan).
Arsitektur V1 dibangun menggunakan *ESP32-Devkit V1* dan modul *off-the-shelf* yang dihubungkan pada PCB THT.	*Dev-kit* memiliki banyak komponen yang tidak diperlukan dan terus menguras daya, seperti: ● Regulator LDO AMS1117 yang tidak efisien dengan *quiescent current* tinggi. ● Chip USB-to-Serial dan LED indikator yang terus aktif.

V1 menggunakan mode *Deep Sleep* ESP32. Tugas penghitungan *pulse* curah hujan dikerjakan oleh *ULP (Ultra-Low-Power) Co-Processor* internal.	Kegagalan *Deep Sleep* ini adalah kesalahan desain fatal. Untuk memonitor *pulse*, ULP Co-Processor harus aktif hampir sepanjang waktu (*duty cycle* 4ms). Akibatnya, *node* tidak pernah benar-benar tidur, menggagalkan tujuan *deep sleep* dan menjadi sumber pemborosan daya *standby* terbesar.
Power gating diimplementasikan, namun secara terbatas hanya pada modul GPS.	Modul LoRa dan semua sensor I2C tidak di *power gate*. Komponen ini tetap terhubung ke daya selama *deep sleep*, menyebabkan konsumsi arus yang signifikan dan konstan.
Frekuensi CPU hanya dapat diturunkan ke 80MHz. *Firmware* juga memiliki fitur OTA dan Web Config saat *booting*.	Frekuensi 80MHz masih terlalu tinggi dan boros daya untuk tugas ringan seperti membaca sensor. Selain itu, fitur OTA/Web Config yang aktif 20 detik saat *boot* menghabiskan daya dalam jumlah besar secara percuma karena harus menyalakan Wi-Fi.

Berdasarkan analisis kegagalan *node* V1, perancangan ulang *node* V2 difokuskan untuk mengatasi setiap sumber inefisiensi tersebut. Fokus perbaikan untuk arsitektur *node* V2 adalah sebagai berikut:

- Mengganti regulator LDO V1 (AMS1117) yang boros daya dengan LDO (MIC5219-3.3V) berdaya sangat rendah, yang memiliki *quiescent current* di level *nanoampere* (nA), sehingga meminimalkan konsumsi daya parasitik dari sirkuit *always-on*.
- Mengganti *linear charger* (CN3065) yang tidak efisien dengan IC MPPT *solar charger* (CN3791) untuk memaksimalkan efisiensi

pemanenan daya, terutama dalam kondisi cahaya rendah atau berawan.

- Mengganti mikrokontroler ESP32 *general-purpose* dengan MCU *ultra-low-power* (STM32U585) dan merancang PCB kustom terintegrasi untuk mengeliminasi total komponen parasitik dari *dev-kit*.
- Mengganti konsep *deep sleep* V1 yang gagal dengan arsitektur *true power-off*. Ini dicapai dengan menggunakan timer eksternal TPL5111 dan *load switch* TPS22918. Saklar ini, akan memutus total suplai daya ke *semua peripheral* boros daya (termasuk MCU, modul LoRa, dan sensor I2C) selama *node* pada fase *standby*.
- Melakukan *offloading* tugas *always-on*. Tugas krusial penghitungan *pulse* hujan dipindahkan dari ULP MCU ke sirkuit *counter* eksternal berbasis CD4040BM berdaya sangat rendah yang dirancang khusus untuk tetap aktif di *always-on region*.

2.2 Tinjauan Teori

2.2.1 Internet of Things

Internet of Things (IoT) adalah jaringan perangkat fisik yang saling terhubung melalui internet untuk mengumpulkan, berbagi, dan memproses data secara otomatis. Perangkat IoT biasanya dilengkapi dengan sensor, aktuator, dan sistem komunikasi untuk mendukung interaksi antar perangkat dan pengguna (Gubbi et al., 2013). IoT telah diimplementasikan di berbagai bidang, seperti industri, kesehatan, rumah pintar, dan pertanian, dengan tujuan meningkatkan efisiensi dan mengurangi intervensi manusia. [5]

Menurut Atzori et al. (2010), IoT mengintegrasikan tiga konsep utama, yaitu: [6]

1. **Sensor dan Aktuator**

   Perangkat untuk mendeteksi perubahan fisik di lingkungan dan melakukan tindakan berdasarkan data yang diperoleh.

2. **Jaringan Komunikasi**

   Media untuk mentransfer data antar perangkat, seperti WiFi, LoRa, ZigBee, atau jaringan seluler.

3. **Pemrosesan Data**

   Analisis dan pengelolaan data untuk menghasilkan informasi yang dapat ditindaklanjuti.

IoT telah banyak diterapkan dalam sektor pertanian untuk meningkatkan efisiensi produksi, memonitor kondisi lingkungan, dan mengoptimalkan penggunaan sumber daya. Teknologi IoT memungkinkan petani untuk memantau kondisi tanah, cuaca, dan kesehatan tanaman secara real-time menggunakan perangkat sensor yang terhubung ke jaringan komunikasi (Patel, 2016). [7]

Namun, penerapan IoT di lokasi terpencil seperti perkebunan salak menghadapi tantangan unik, seperti:

- **Keterbatasan Infrastruktur Jaringan**

  Tidak adanya akses internet atau WiFi di kebun membuat metode komunikasi berbasis internet seperti MQTT atau cloud sulit diterapkan.

- **Jarak yang Jauh Antar Node**

Perangkat IoT di kebun sering kali tersebar dengan jarak antar node mencapai ratusan meter, yang tidak dapat dicapai oleh teknologi komunikasi jarak pendek seperti WiFi atau Bluetooth.

Untuk mengatasi keterbatasan ini, teknologi jaringan LoRa sering dipilih sebagai solusi utama karena kemampuan jangkauan jarak jauhnya (hingga 10 km dalam kondisi optimal) dan efisiensinya dalam penggunaan daya (Sinha et al., 2017). [8]

2.2.2 LoRa

LoRa (Long Range) adalah teknologi komunikasi nirkabel jarak jauh yang menggunakan modulasi chirp spread spectrum (CSS) untuk mentransmisikan data pada pita frekuensi ISM (Industrial, Scientific, and Medical). LoRa dirancang khusus untuk aplikasi IoT yang membutuhkan jangkauan luas dan konsumsi daya rendah. Teknologi ini menjadi salah satu teknologi utama dalam jaringan LPWA (Low Power Wide Area) karena efisiensinya dalam menghubungkan perangkat-perangkat IoT di lokasi terpencil tanpa infrastruktur komunikasi yang mahal (Sinha et al., 2017). [8]

LoRa menggunakan modulasi CSS untuk mentransmisikan data dalam bentuk sinyal frekuensi yang berubah secara bertahap (chirp). Keunggulan utama dari modulasi ini adalah kemampuan untuk mempertahankan integritas data meskipun dalam kondisi interferensi atau sinyal lemah. LoRa beroperasi pada frekuensi ISM (seperti 433 MHz, 868 MHz, dan 915 MHz) yang tidak memerlukan lisensi. [9]

Modul Transceiver LoRa SX1276 adalah komponen perangkat keras inti dalam bentuk sirkuit *Integrated Circuit* (IC) yang mengimplementasikan teknologi komunikasi *wireless* LoRa. IC ini bertanggung jawab untuk

proses modulasi, transmisi, dan penerimaan sinyal radio. Fitur-fitur utamanya yang relevan untuk aplikasi IoT di Indonesia meliputi:

- Beroperasi pada pita frekuensi 920-923 MHz, sesuai dengan regulasi yang ditetapkan oleh PM Kominfo No.1 Tahun 2019 dan PERDIRJEN SDPPI No. 3 Tahun 2019.
- Tingkat *transmit power* dapat dikonfigurasi melalui perangkat lunak, memungkinkan penyesuaian antara jangkauan sinyal dan konsumsi daya.

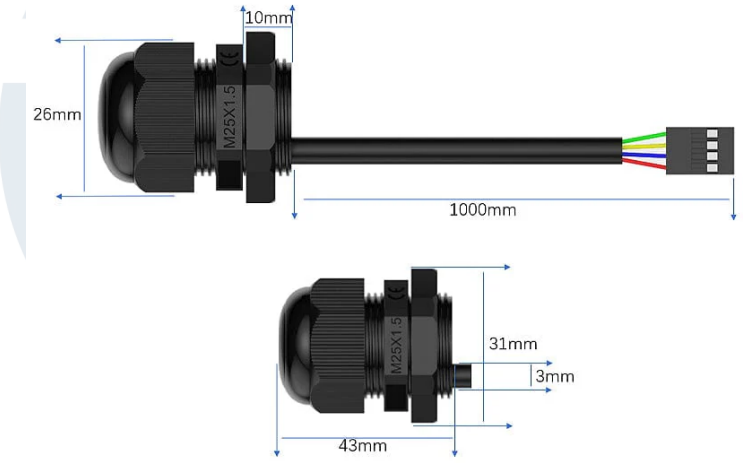
2.2.3 Sensor Suhu dan Kelembapan (DFRobot SHT20)

Gambar 2.1 Sensor SHT20

Sensor SHT20 adalah sebuah sensor digital berbasis dari 4C CMOSens® presisi yang berfungsi untuk mengukur suhu dan kelembapan relatif (RH). Sensor ini beroperasi pada tegangan 3.3V dan berkomunikasi dengan mikrokontroler melalui *interface* I2C. Berdasarkan datasheet-nya, SHT20 menawarkan tingkat akurasi ±0.3°C untuk suhu dan ±3% RH untuk kelembapan, dengan resolusi

hingga 14-bit. Sensor ini meiliki IP68 rating sehingga dikategorikann *weatherproof*. Konsumsi dayanya yang rendah (*claim* maksimum 350µA saat start-up) membuatnya sangat ideal untuk aplikasi IoT bertenaga baterai. [10]

2.2.4 Sensor *Ambient Light*



Gambar 2.2 Sensor DFRobot Ambient Light

Sensor DFRobot Gravity Ambient Light yang berbasis BH1750 adalah sensor digital untuk mengukur tingkat iluminasi cahaya dalam satuan Lux. Sama seperti SHT20, sensor ini menggunakan antarmuka komunikasi I2C. Sensor ini memiliki respons spektral yang mendekati persepsi mata manusia (*photopic response*) dengan rentang deteksi dari 1 Lux hingga 65535 Lux dan akurasi ±1.2 Lux. Konsumsi daya dari sensor ini diklaim hanya menggunakan 1µA saat pengoperasiannya, sehingga sensor ini sangat efisien dalam daya. [11]

2.2.5 Tipping Bucket Rain Gauge

Sensor curah hujan yang digunakan bekerja dengan mekanisme *tipping bucket*. Air hujan akan dikumpulkan hingga wadah internal terjungkit (*tipping*) setiap mencapai volume setara 0.47 mm curah hujan. Setiap jungkitan akan membuat magnet kecil bergerak melewati sensor *Hall-effect switch* yang beroperasi pada tegangan 3.3V, yang kemudian menghasilkan sebuah *pulse* listrik sesaat. Jumlah *pulse* inilah yang dihitung untuk menentukan total curah hujan.

2.2.6 Mikrokontroler STM32U585 [12]

STM32U585 adalah mikrokontroler 32-bit dari STMicroelectronics yang dirancang secara spesifik untuk aplikasi ultra-low-power (ULP). Berbasis ARM Cortex-M33, MCU ini dibangun di atas proses fabrikasi 40nm yang sangat efisien. Berbeda dengan mikrokontroler serba guna seperti ESP32, seri STM32U5 tidak menyertakan modul radio boros daya yang tidak diperlukan untuk aplikasi LoRa murni, seperti Wi-Fi atau Bluetooth Classic, sehingga secara fundamental menghilangkan sumber konsumsi daya parasitik.

2.2.7 STM32duino [13]

STM32duino merupakan sebuah proyek open-source yang menyediakan lapisan kompatibilitas (compatibility layer) untuk memungkinkan pengembangan perangkat lunak bagi keluarga mikrokontroler STMicroelectronics STM32 dengan menggunakan API dan *development environment* Arduino. Tujuan utama dari proyek ini adalah untuk menyederhanakan proses prototipe pada

platform perangkat keras yang canggih dengan memungkinkan pengembang untuk memanfaatkan *syntax* dan library yang familiar dari ekosistem Arduino. Dengan demikian, STM32duino secara efektif menurunkan kurva pembelajaran yang umumnya diasosiasikan dengan *development environment* STM32 native seperti STM32CubeIDE atau Keil tanpa mengorbankan akses terhadap performa dan fitur superior yang ditawarkan oleh mikrokontroler STM32.

Secara fundamental, arsitektur STM32duino bekerja dengan cara menerjemahkan panggilan fungsi tingkat tinggi dari API Arduino menjadi fungsi-fungsi yang relevan dalam Hardware Abstraction Layer (HAL) yang disediakan secara resmi oleh STMicroelectronics. STM32Cube HAL adalah sebuah library driver level rendah yang menstandarkan interaksi dengan *peripheral* perangkat keras seperti GPIO, I2C, dan SPI di seluruh varian mikrokontroler STM32. Dalam praktiknya, core STM32duino berfungsi sebagai lapisan perantara yang mengonversi perintah sederhana seperti digitalWrite() menjadi panggilan fungsi HAL yang lebih kompleks, contohnya HAL_GPIO_WritePin(). Arsitektur berlapis ini memberikan keuntungan seperti kemudahan dan kecepatan pengembangan dari framework Arduino, sambil tetap mengandalkan eksekusi level rendah yang teroptimasi dan efisien dari driver perangkat keras resmi yang disediakan oleh produsen chip.

2.2.8 Arduino *Framework* [14]

Arduino framework adalah sebuah platform elektronika open-source yang dirancang untuk menyederhanakan proses pemrograman mikrokontroler. Fondasinya dibangun di atas bahasa pemrograman C/C++ yang disederhanakan, yang distrukturkan ke dalam dua fungsi

utama: setup(), yang dieksekusi satu kali saat perangkat dinyalakan untuk inisialisasi, dan loop(), yang dieksekusi secara berulang-ulang sebagai program utama. Kerangka kerja ini menyediakan serangkaian library inti yang kaya, yang menawarkan Antarmuka Pemrograman Aplikasi (API) tingkat tinggi untuk mengontrol fungsionalitas perangkat keras yang kompleks. Perintah-perintah yang umum dikenal seperti digitalWrite(), analogRead(), dan Serial.print() merupakan bentuk abstraksi dari operasi-operasi level register yang rumit, sehingga memungkinkan pengembang untuk berinteraksi dengan pin GPIO, ADC, dan komunikasi serial tanpa perlu memahami arsitektur mikrokontroler secara mendalam.

Kekuatan utama dari Arduino framework terletak pada lapisan abstraksi perangkat kerasnya yang konsisten, yang memungkinkan portabilitas kode di antara berbagai arsitektur mikrokontroler yang berbeda. Dengan adanya core spesifik untuk setiap platform (seperti AVR, ESP32, atau STM32), kode yang sama dapat dikompilasi dan dijalankan dengan modifikasi minimal, meskipun perangkat keras dasarnya sangat bervariasi. Ekosistem ini diperkaya lebih lanjut oleh *library manager* yang menyediakan akses ke ribuan *library* pihak ketiga untuk berbagai sensor, aktuator, dan protokol komunikasi. Secara keseluruhan, Arduino framework menggabungkan compiler toolchain yang sudah dikonfigurasi sebelumnya dengan API yang sederhana dan ekosistem library yang luas, menciptakan sebuah lingkungan pengembangan yang sangat mudah diakses namun tetap cukup kuat untuk pengembangan produk yang kompleks.

2.2.9 Prinsip Clock Gating dan Power Gating [16]

Salah satu tinjauan komprehensif mengenai teknik-teknik manajemen daya pada level sistem elektronik disajikan oleh Barge

dan Mary (2024). Dalam penelitian mereka, dipaparkan berbagai metode yang dapat diterapkan mulai dari level desain sirkuit terintegrasi (IC) hingga level sistem. Meskipun beberapa teknik menargetkan desainer *chip* (SoC/FPGA), prinsip-prinsip dasarnya sangat relevan dan dapat diadopsi pada desain *board-level* untuk *IoT node* di MySalak V2. Penelitian ini menggarisbawahi beberapa pendekatan kunci, di antaranya adalah *Clock Gating dan Power Gating*.

Clock Gating adalah sebuah teknik optimasi daya yang fundamental pada level desain sirkuit digital, yang bertujuan untuk mengurangi konsumsi daya dinamis. Konsep dasarnya adalah menonaktifkan *clock* ke blok-blok sirkuit atau register yang sedang tidak aktif atau tidak diperlukan untuk sebuah operasi. Karena sebagian besar konsumsi daya dinamis pada sebuah chip CMOS terjadi saat adanya transisi sinyal (dari 0 ke 1 atau sebaliknya) yang dipicu oleh *clock*, maka dengan menghentikan *clock* ke sirkuit yang menganggur sehingga aktivitas yang tidak perlu dapat dieliminasi.

Power Gating adalah teknik manajemen daya yang lebih agresif dibandingkan *clock gating*, yang dirancang untuk menekan konsumsi daya statis atau arus bocor (*leakage current*). Prinsip kerjanya adalah dengan menggunakan sebuah saklar transistor (sering disebut *sleep transistor*) untuk memutus sepenuhnya hubungan antara suplai VDD dan *ground* dari sebuah blok sirkuit saat blok tersebut tidak digunakan dalam periode yang lama. Dengan memutus total sumber dayanya, blok sirkuit tersebut tidak akan mengonsumsi daya sama sekali, termasuk arus bocor yang tetap ada meskipun sinyal *clock* sudah dimatikan.

2.2.10 Prinsip *Duty Cycling* dan *Sleep Scheduling* [17]

Seperti yang dibahas oleh Manohar dan Dharini (2024), menegaskan bahwa *sleep scheduling* dan *duty cycling* adalah strategi yang paling krusial untuk manajemen daya pada *node* sensor *wireless*. Mengingat perangkat IoT di lapangan sangat bergantung pada sumber daya baterai yang terbatas, memperpanjang masa pakai operasional menjadi tantangan utama. Konsep *duty cycling* secara langsung menjawab tantangan ini dengan mengubah paradigma operasional perangkat dari "selalu aktif" menjadi siklus periodik yang didominasi oleh mode tidur berdaya rendah. Perangkat hanya diaktifkan untuk interval waktu yang sangat singkat guna melakukan tugas-tugas esensial seperti akuisisi data sensor, pemrosesan, dan transmisi nirkabel sebelum segera kembali ke kondisi tidur untuk periode yang jauh lebih lama.

Implementasi *duty cycling* adalah alasan utama mengapa penggunaan mode *deep sleep* atau *standby* menjadi sangat penting. Dengan membuat *node* menghabiskan 99.9% waktunya dalam kondisi tidur, konsumsi daya rata-rata dapat ditekan hingga mendekati level mikroampere, yang secara teoritis dapat memperpanjang masa pakai baterai dari hitungan hari menjadi bulan atau bahkan tahun.

Seluruh logika perangkat lunak pada MySalak V2 akan dirancang di sekitar siklus ini. Ini memaksa perancangan *firmware* yang ringkas, di mana setiap milidetik waktu aktif menjadi sangat berharga dan harus dioptimalkan. Strategi *duty cycling* memperkuat keputusan untuk memilih komponen dengan waktu respons yang cepat dan konsumsi *quiescent current* yang sangat rendah. Karena sebagian besar waktu dihabiskan dalam mode tidur, maka efisiensi komponen

pada saat tidak aktif menjadi sama pentingnya dengan efisiensi saat aktif.

2.2.11 Prinsip Ketahanan Daya (*Power Budget*)

Ketahanan daya pada sistem IoT *self-powered* adalah analisis keseimbangan daya yang bertujuan untuk memastikan keberlanjutan operasional perangkat dalam jangka panjang. Prinsip ini tidak hanya berfokus pada seberapa besar kapasitas baterai yang tersedia, melainkan pada interaksi dinamis antara tiga variabel utama yaitu daya yang dipanen (*power harvested*), daya yang dikonsumsi (*power consumed*), dan kapasitas penyimpanan (*power storage*).

Dalam aplikasinya, *Power Budget* dapat dikategorikan menjadi dua kondisi:

1. ***Positive Power Budget***

   Kondisi ideal di mana rata-rata daya harian yang dipanen melebihi rata-rata daya yang dikonsumsi ($P_{harvested} > P_{consumed}$). Kelebihan daya ini akan disimpan dalam baterai hingga mencapai kapasitas penuh, memungkinkan sistem untuk bertahan hidup melewati periode tanpa matahari (malam hari atau cuaca mendung) tanpa menguras baterai hingga batas kritis.

2. ***Negative Power Budget***

   Kondisi kritis di mana konsumsi daya melebihi kemampuan pemanenan ($P_{harvested} < P_{consumed}$). Dalam kondisi ini, sistem akan terus-menerus menguras cadangan baterai hingga mencapai ambang batas *cut-off*.

Namun, perlu ditekankan bahwa dalam implementasi MySalak, tujuan utama *power budget* tidak harus selalu mencapai kondisi surplus daya setiap saat, mengingat kondisi lingkungan perkebunan yang rimbun. Sasaran kritisnya adalah memaksimalkan peran $P_{harvested}$ untuk menekan pengurasan daya dari baterai seminimal mungkin. Meskipun daya yang dipanen mungkin tidak selalu lebih besar dari yang dikonsumsi (terutama saat cuaca buruk), kontribusinya harus cukup signifikan untuk memperlambat laju penurunan tegangan baterai secara drastis. Dengan demikian, strategi ini bertujuan untuk memperpanjang masa pakai alat (*uptime*) selama mungkin hingga melampaui jadwal pemeliharaan berkala, menjamin keberlanjutan sistem operasional meskipun tidak mencapai kemandirian daya.