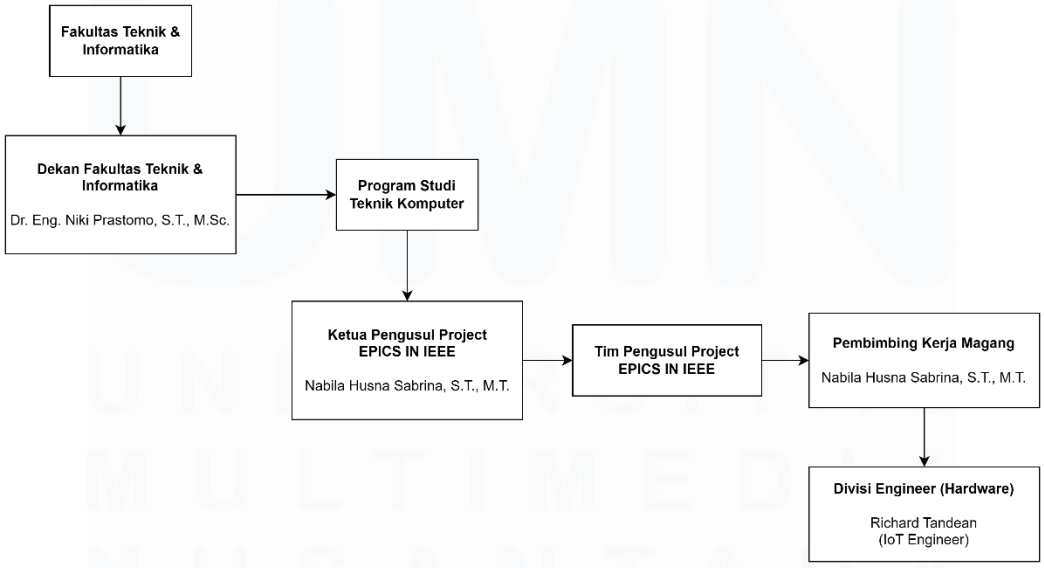
BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Pada pelaksanaan kerja magang ini, penulis terlibat dalam *project* EPICS (*Engineering Projects in Community Service*). Program ini bertujuan untuk menyediakan organisasi pelayanan masyarakat dengan teknologi yang mereka butuhkan untuk meningkatkan layanan kepada masyarakat, serta memberikan pengalaman belajar kepada mahasiswa untuk memperluas keterampilan mereka. [2]

Pada *project* EPICS ini, penulis beserta anggota *project* lainnya bekerja sama dengan Paguyuban Mitra Turindo dan Universitas Gadjah Mada (UGM). *Project* EPICS IN IEEE yang akan dilaksanakan yaitu *An AIoT-powered Smart Agricultural System for Pets Forecasting and Management*. Penulis tergabung di divisi *engineer* sebagai IoT Engineer di bawah bimbingan Bu Nabila Husna Sabrina. Dalam *project* ini juga penulis nantinya akan bekerja sama dengan *Artificial Intellegence engineer* dan *Web Developer* untuk mengintegrasikan data dari sensor ke *web app* yang nantinya akan digunakan oleh user (petani salak). Kedudukan pada pelaksanaan kerja magang ini dapat dilihat di gambar 3.1.1.



Gambar 3.1 Kedudukan posisi pemagang

3.2 Tugas dan Uraian Kerja Magang

3.2.1 Tugas Kerja Magang

Tabel 3.1 Timeline Kerja Magang

Februari	Minggu 1: • Mencari beberpaa hardware yang cocok untuk digunakan pada *prototype*. • Melakukan pemilihan sensor dari spesifikasi & kebutuhan. Minggu 2: • Mencari *library* yang bisa digunakan untuk *hardware* sensor yang telah dipilih. • Mencoba *hardware* sensor yang telah dipilih. Minggu 3: • Mencoba *hardware* sensor yang telah dipilih. Minggu 4: • Survey lokasi kebun salak Mitra Turindo. • Studi banding dengan UGM. • Mapping kebun salak.
Maret	• Melakukan LoRa *Testing*
April	Minggu 1: • Mengukur jarak jangkauan LoRa Minggu 2: • Menggabungkan sensor – sensor yang digunakan Minggu 3: • Melakukan validasi data sensor Minggu 4: • Melakukan validasi data sensor
Mei	Mengerjakan laporan magang

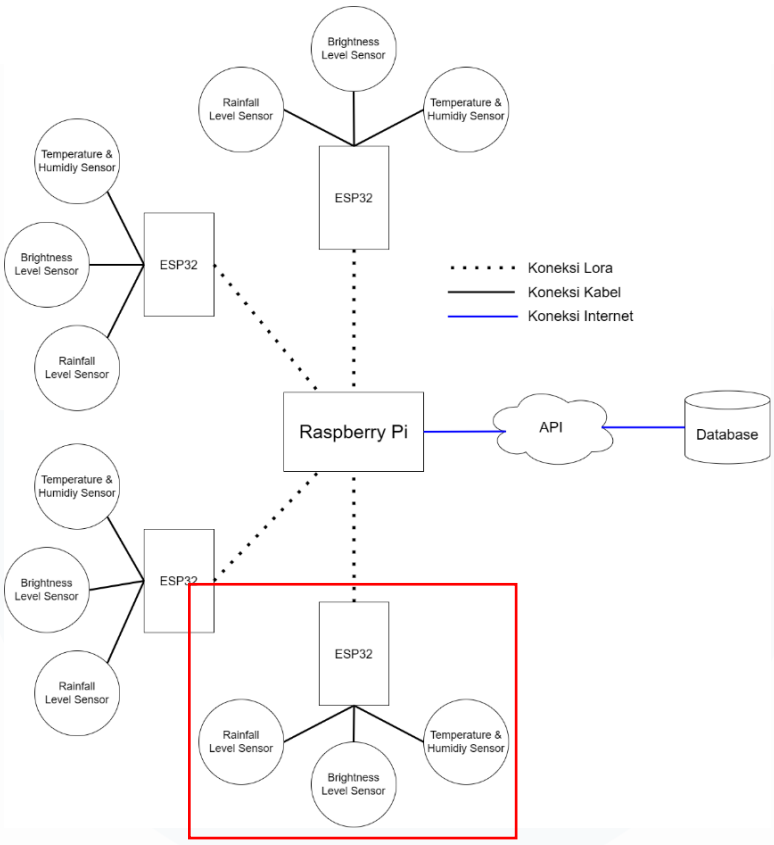
Pada *project* ini, penulis ditugaskan untuk membuat *prototype* hardware yang nantinya akan digunakan untuk membantu para petani salak di Paguyuban Mitra Turindo memantau kondisi cuaca secara *realtime* untuk mencegah atau mengurangi serangan hama pada buah salak. Setelah menggali informasi dari petani di Paguyuban Mitra Turindo ada 2 faktor yang dapat memicu munculnya hama, yaitu kondisi cuaca dan tingkat curah hujan. Maka dari itu, *prototype* yang akan dibuat harus bisa mengambil data seperti tingkat keterangan cahaya (*brightness level*), suhu / temperatur udara (*air temperature*), tingkat kelembapan udara (*air humidity*) dan tingkat curah hujan (*rainfall level*).

Sebelum pembuatan *prototype*, penulis terlebih dahulu memastikan hal-hal yang penting pada *prototype* seperti tujuan *prototype*, keadaan pada lokasi penempatan *prototype* dan data apa saja yang diperlukan pada *web app* yang akan diimplementasikan nantinya. Penulis kemudian melakukan riset terhadap beberapa sensor yang sekiranya cocok secara spesifikasi dan kondisi yang ada pada lokasi penempatan *prototype* nantinya.

Setelah mengumpulkan sensor dan barang – barang yang diperlukan, penulis memulai untuk mencoba untuk mengambil data dari sensor – sensor yang telah dipilih. Lalu setelah berhasil mengambil data dari setiap sensor, penulis melakukan validasi data dari sensor yang digunakan dengan menggunakan sensor yang memiliki akurasi kurang lebih sama dengan sensor yang digunakan.

3.2.2 Uraian Kerja Magang

Pada awal pembuatan *prototype*, penulis terlebih dahulu membuat diagram blok sistem yang nantinya akan diimplementasikan pada *project* EPICS IN IEEE. Berikut diagram blok sistem secara keseluruhan:



Gambar 3.2 Diagram Block Sistem *Project* IEEE

Seperti yang dapat dilihat pada gambar blok diagram sistem diatas, nantinya akan digunakan 1 Raspberry Pi yang akan digunakan sebagai *server*, 4 ESP32 yang akan digunakan sebagai *node* dan 3 buah sensor yang akan disambungkan pada mikrokontroler ESP32 yaitu sensor suhu dan kelembaban udara, sensor cahaya dan sensor curah hujan. *Prototype* yang akan dibuat akan terdiri dari 1 ESP32 sebagai mikrokontroler, 1 sensor suhu dan kelembaban, 1 sensor keterangan cahaya, 1 sensor curah hujan, 1 module LoRa dan solar panel beserta modulnya.

Setelah merancang sistem yang akan diimplemantasikan, penulis ditugaskan untuk mencari komponen *hardware* dan sensor yang cocok untuk digunakan untuk digunakan pada *prototype*. *Prototype* tersebut mencakup ke dalam kotak merah pada gambar 3.2.2.1. Berikut daftar komponen *hardware* yang akan digunakan pada *prototype*:

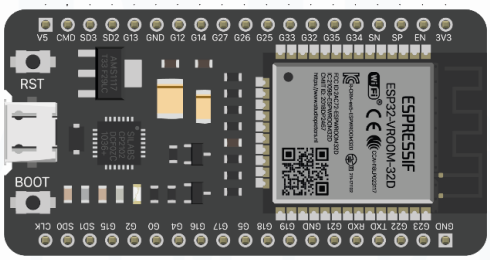
1. Raspberry Pi

   Alasan penulis memilih Raspberry Pi sebagai sebagai *server* adalah karena Raspberry Pi memiliki GPIO untuk menjadi LoRa *Receiver*, memiliki banyak dokumentasi sehingga akan mempermudah pengembangan sistem, lebih cocok buat dijadikan server karena kapasitas komputasi dan memori RAM yang lebih besar. Raspberry Pi yang digunakan adalah Raspberry Pi Model 3 B seperti yang ada pada gambar 3.3.

Gambar 3.3 Mikrokontroller Raspberry Pi Model 3

2. ESP32 (Mikrokontroler)

   Alasan penulis memilih ESP32 sebagai kontroller utama *prototype* adalah karena ESP32 memiliki harga yang terjangkau untuk sebuah mikrokontroler, ESP32 kompatibel untuk digunakan di ArduinoIDE dan MicroPython, ESP32 memiliki banyak dokumentasi yang dapat mempermudah dalam mengembangkan *prototype* dan ESP32 juga bisa menggunakan jalur komunikasi I2C yang dibutuhkan oleh sensor.



Gambar 3.4 Mikrokontroller ESP32

3. DFRobot SHT20 *Temperature & Humidity Sensor*

   DFRobot SHT20 *Temperature & Humidity Sensor* adalah sensor yang digunakan untuk mengukut suhu serta kelembaban udara disekitar. Alasan penulis memilih ini adalah karena sensor ini memiliki akurasi yang tinggi sehingga bisa menghasilkan data yang akurat dan sensor ini juga sudah dirancang agar bisa digunakan di luar ruangan karena *waterproof*. Gambar dan spesifikasi sensor dapat dilihat pada gambar 3.5 dan tabel 3.2. [3]

Gambar 3.5 DFRobot SHT20 Temperature & Humidity Sensor

Tabel 3.2 Spesifikasi DFRobot SHT20 Temperature & Humidity Sensor

Operating Voltage	3.3/5V
Communication Interface	I2C
Accuracy	±3% RH / ±0.3 ℃
Measuring Range	0-100% RH / -40-125 ℃

4. DFRobot *Gravity I2C IP68 Waterproof Ambient Light Sensor*

   DFRobot *Gravity I2C IP68 Waterproof Ambient Light Sensor* adalah sensor yang digunakan untuk mengukur keterangan cahaya pada lingkungan sensor. Alasan penulis memilih ini adalah karena sensor ini memiliki akurasi yang tinggi sehingga bisa menghasilkan data yang akurat dan sensor ini juga sudah dirancang agar bisa digunakan

di luar ruangan karena *waterproof*. Gambar dan spesifikasi sensor dapat dilihat pada gambar 3.6 dan 3.3 [4]

Gambar 3.6 DFRobot *Gravity I2C IP68 Waterproof Ambient Light Sensor*

Tabel 3.3 Spesifikasi DFRobot Gravity I2C IP68 Waterproof Ambient Light Sensor

Operating Voltage	5V
Communication Interface	I2C
Accuracy	1.2 lux
Measuring Range	1 – 65535 lux

5. Nayyara *Ombrometer Tipping Bucket Rain Gauge*

   Nayyara *Ombrometer Tipping Bucket Rain Gauge* adalah sensor tipping bucket yang memakai hall effect sensor A3144 untuk menghitung jumlah tip setiap kali air hujan masuk ke sensor. Sensor ini bekerja dengan menggunakan sensor magnetic dimana jika satu sisi tip bucket penuh, maka tip tersebut akan menumpahkan airnya dan besi magnet yang tertempel pada tip akan melewati sensor tersebut. Jika ada magnet yang melewati sensor tersebut, maka interrupt akan menambah jumlah counter pada ESP32. Nantinya total counter tip akan dikalkulasi jumlahnya perhari dan akan dikalikan 0,47 ml (jumlah maksimal air yang dapat ditampung satu sisi tipping bucket). Alasan penulis memilih Nayyara *Ombrometer Tipping Bucket Rain*

Gauge Sensor adalah karena sensor ini bisa dipakai pada *prototype* karena bentuknya yang dirancang dengan menggunakan 2 lubang untuk dipasangkan baut sehingga bisa digunakan pada *prototype*. Selain itu sensor ini juga telah tersedia perhitungan kalibrasi yang dilakukan oleh penjual untuk mengetahui jumlah air yang tertampung pada setiap tipnya. Gambar dan spesifikasi sensor dapat dilihat pada gambar 3.7 dan tabel 3.4.

Gambar 3.7 Nayyara Ombrometer Tipping Bucket Rain Gauge

Tabel 3.4 Spesifikasi Nayyara *Ombrometer Tipping Bucket Rain Gauge*

Operating Voltage	3.3/5V
Communication Interface	Serial
Mililiter per tip	0,47 ml

6. Cosmic.id LoRa RFM95

   Cosmic.id LoRa RFM95 adalah module LoRa yang memungkinkan perangkat IoT berkomunikasi secara jarak jauh dengan menggunakan modulasi radio namun tetap dengan konsumsi daya yang rendah dan merupakan *physical layer protocol* karena sinyal yang ditransmisikan merupakan sinyal fisik. Lora Alasan penulis memilih Cosmic.id LoRa RFM95 adalah karena module ini memiliki banyak library yang dapat mempermudah penggunaan LoRa. Selain itu, penulis menggunakan LoRa sebagai alat komunikasi dibanding dengan alat komunikasi yang lain seperti WiFi adalah karena pada implementasi prototype nanti jarak antar Raspberry Pi dan ESP32 sangat jauh sekitar 2 – 3 km

dan juga keterbatasan kesediaan listrik pada lokasi penempatan prototype. Maka dari itu diperlukan alat komunikasi yang cocok untuk pengiriman data jarak jauh dan hemat daya seperti LoRa. Gambar dan spesifikasi sensor dapat dilihat pada gambar 3.8 dan tabel 3.5.

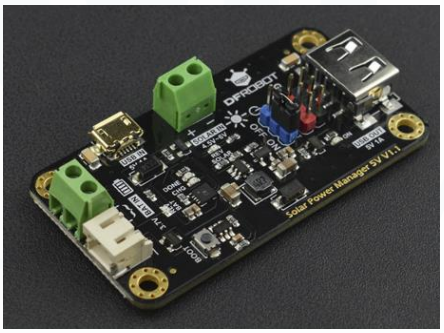
Gambar 3.8 Cosmic.id LoRa RFM95

Tabel 3.5 Spesifikasi Cosmic.id LoRa RFM95

Operating Voltage	3.3 - 6V
Frequency Range	908 – 928Mhz

7. DFRobot Solar Panel Manager 5V

   DFRobot Solar Panel Manager 5V adalah module yang dipakai untuk mengatur penggunaan solar panel agar dapat digunakan pada *prototype*. Gambar dan spesifikasi sensor dapat dilihat pada gambar 3.9 dan tabel 3.6. [5]



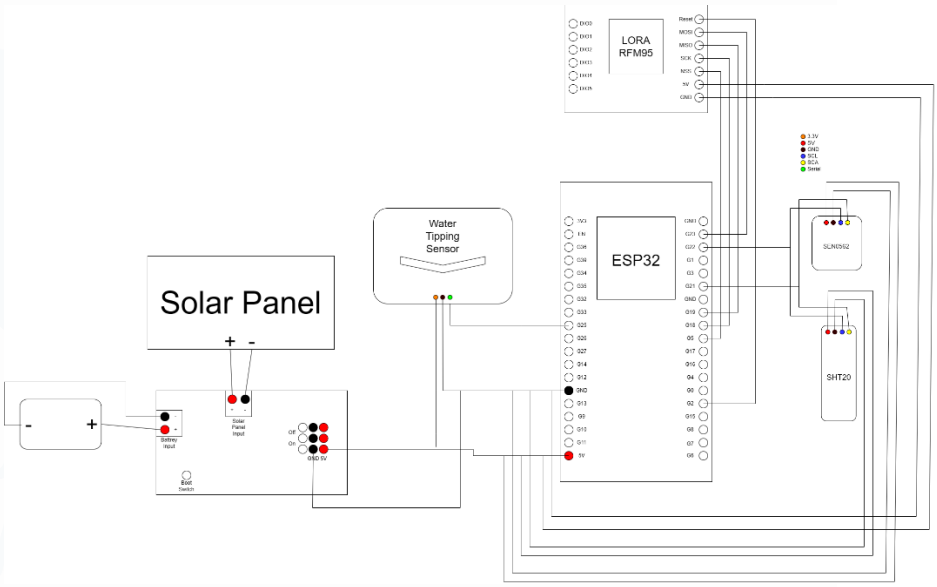
Gambar 3.9 DFRobot Solar Panel Manager 5V

Tabel 3.6 Spesifikasi DFRobot Solar Panel Manager 5V

Voltage Input	4.4 – 6V
Battrey Input	3.7V
Voltage Output	5V
Operation Temperature	-40℃ - 85℃

Hardware yang dipilih diatas merupakan *hardware* pilihan yang memiliki spesifikasi yang cocok dengan yang diperlukan pada prototype yang akan diimplementasikan. Sensor – sensor yang dipilih penulis juga telah memiliki akurasi yang tinggi berdasarkan datasheet pada masing – masing sensor dibandingkan dengan beberapa sensor yang menjadi pertimbangan pada saat melakukan pemilihan sensor.

Setelah menentukan komponen yang akan digunakan, penulis merancang wiring diagram yang akan diimplementasikan pada *prototype*. *Wiring diagram* yang akan diimplementasikan dapat dilihat pada gambar 3.10 dan *pinout* masing – masing *hardware* dapat dilihat pada tabel 3.7.



Gambar 3.10 Wiring Diagram *Prototype* ESP32

Tabel 3.7 Wiring Diagram *Prototype* ESP32

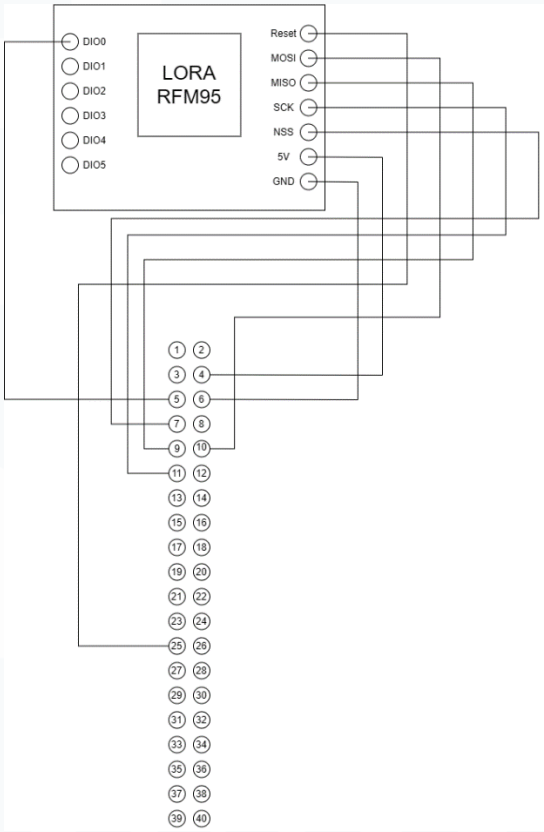
DFRobot *SHT20 Temperature & Humidity Sensor*	
SCL	GPIO22
SCA	GPIO21
DFRobot *Gravity I2C IP68 Waterproof Ambient Light Sensor*	
SCL	GPIO22
SCA	GPIO21
Nayyara *Ombrometer Tipping Bucket Rain Gauge*	
Serial	GPIO25
Cosmic.id LoRa RFM95	
DIO0	GPIO4
Rset	GPIO15
MOSI	GPIO23
MISO	GPIO19
SCK	GPIO18
NSS	GPIO5

Gambar 3.11 Implementasi *Prototype* ESP32

Penulis merancang wiring diagram seperti pada gambar 3.2.2.8 dengan alasan yaitu:

1. DFRobot I2C *Temperature & Humidity Sensor*:
   - SCL (*Serial Clock*) dihubungkan ke GPIO22 karena pin ini digunakan khusus untuk mengirimkan sinyal clock pada protokol I2C dan merupakan pin *default* SCL pada ESP32.
   - SDA (*Serial Data*) dihubungkan ke GPIO21 karena pin ini digunakan untuk mengirim dan menerima data pada protokol I2C dan merupakan pin *default* SDA pada ESP32.
2. DFRobot *Gravity* I2C IP68 *Waterproof Ambient Light Sensor*
   - SCL (*Serial Clock*) dihubungkan ke GPIO22 karena pin ini digunakan untuk mengirimkan sinyal clock pada protokol I2C dan merupakan pin *default* SCL pada ESP32
   - SDA (*Serial Data*) dihubungkan ke GPIO21 karena pin ini digunakan untuk mengirim dan menerima data pada protokol I2C dan merupakan pin *default* SDA pada ESP32.
3. Nayyara *Ombrometer Tipping Bucket Rain Gauge*
   - Serial dihubungkan ke GPIO25 karena pin ini cukup fleksibel dan dapat digunakan sebagi input/output pada komunikasi serial.
4. Cosmic.id LoRa RFM95
   - DIO0 dihubungkan ke GPIO4 karena pin ini biasanya digunakan untuk pin interrupt pada ESP32 dan sesuai dengan *library* yang digunakan.
   - Reset dihubungkan ke GPIO15 karena pin ini biasanya digunakan untuk pin interrupt pada ESP32 dan sesuai dengan *library* yang digunakan.
   - MOSI dihubungan ke GPIO23 karena pin ini digunakan untuk mengirimkan data dari *master* ke *slave* pada protokol SPI.
   - MISO dihubungkan ke GPIO19 karena pin ini digunakan untuk menerima data dari *slave* ke *master* pada protokol SPI.

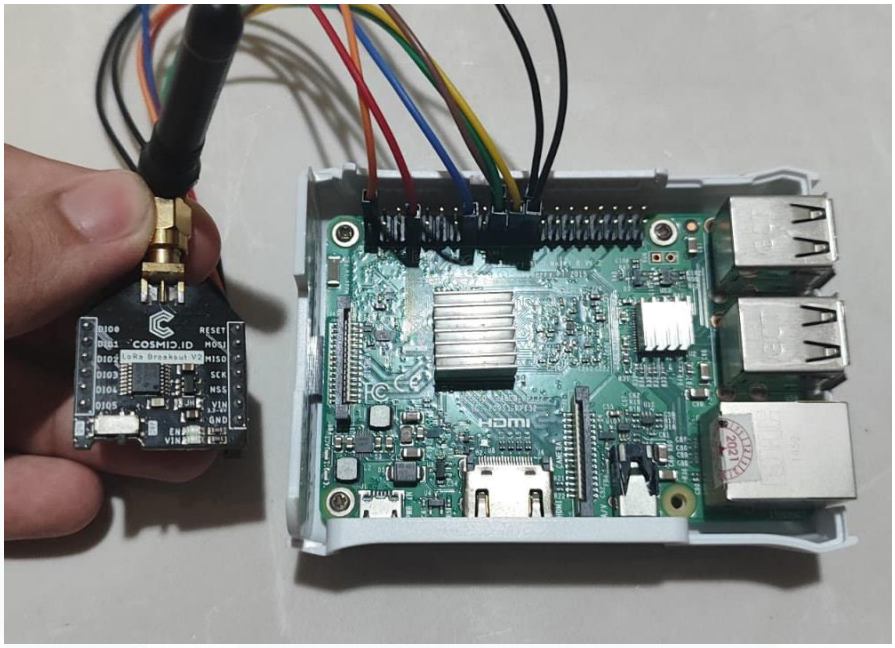
- SCK dihubungkan ke GPIO18 karena pin ini digunakan untuk mengirimkan *clock* pada SPI.
- NSS dihubungkan ke GPIO5 karena pin ini merupakan pin SS (*slave select*) yang digunakan untuk memilih perangkat SPI tertentu pada bus SPI.



Gambar 3.12 Wiring Diagram Raspberry Pi

Tabel 3.8 Wiring Diagram *Prototype* Raspberry Pi

Cosmic.id LoRa RFM95	
DIO0	GPIO5
Reset	GPIO25
MOSI	GPIO10
MISO	GPIO9
SCK	GPIO11
NSS	GPIO7



Gambar 3.13 Implementasi *Prototype* Raspberry Pi

Penulis merancang *wiring diagram* seperti pada gambar 3.2.2.10 dengan alasan berikut:

- DIO0 dihubungkan ke GPIO5: Ini adalah pin yang mungkin digunakan sebagai pin *interrupt* pada Raspberry Pi.
- Reset dihubungkan ke GPIO25: GPIO25 dipilih mungkin karena ketersediaan pin dan kebutuhan fungsi reset.
- MOSI dihubungkan ke GPIO10: Ini adalah pin untuk mengirimkan data dari *master* ke *slave* pada SPI.
- MISO dihubungkan ke GPIO9: Ini adalah pin untuk menerima data dari slave ke *master* pada SPI.
- SCK dihubungkan ke GPIO11: Ini adalah pin untuk mengirimkan *clock* pada SPI.
- NSS dihubungkan ke GPIO7: Ini adalah pin slave select (SS), yang digunakan untuk memilih perangkat SPI tertentu pada bus SPI.

Pada pengembangan *prototype*, penulis ditugaskan untuk melakukan kalibrasi dengan cara menggunakan alat kalibrasi seperti thermometer, kamera termal dan meteran inframerah. Namun dikarenakan keterbatasan alat kalibrasi yang tersedia, penulis mengambil aternatif untuk melakukan validasi data yang dihasilkan oleh sensor yang digunakan. Penulis menggunakan sensor lain yang memiliki akurasi yang sama ataupun mendekati dengan sensor yang nantinya akan dipakai sebagai sensor pada *prototype*. Nantinya dari beberapa sensor yang dicoba akan dilakukan perbandingan apakah data dari sensor – sensor tersebut nilainya mirip atau berjauhan. Berikut beberapa sensor yang penulis gunakan untuk melakukan validasi data sensor:

1. DHT22 (*Temperature & Humidity Sensor*)

Gambar 3.14 Sensor *Temperature & Humidity* DHT22

Tabel 3.9 Spesifikasi DHT22 [6]

Operating Voltage	3.5 – 5.5V
Temperature Range	-40℃ - 80℃
Humidity Range	0% - 100%
Accuracy	0.5℃ & 1%

2. HDC1080 SHT20 (*Temperature & Humidity Sensor*)

Gambar 3.15 *Sensor Temperature & Humidity* HDC1080 SHT20

Tabel 3.10 Spesifikasi HDC1080 SHT20 [7]

Operating Voltage	2.7 – 5.5V
Temperature Range	-20℃ - 85℃
Humidity Range	10% – 80%
Accuracy	0.2 ℃ & 2%

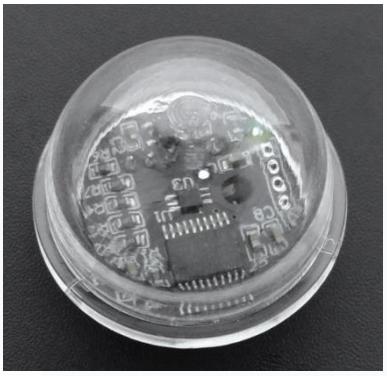
3. DS18B20 (*Temperature Sensor*)

Gambar 3.16 Sensor Temperature DS18B20

Tabel 3.11 Spesifikasi DS18B20 [8]

Operating Voltage	3 – 5.5V
Operating Temperature Range	-50℃ - 125℃
Temperature Range	-10℃ - 85℃
Accuracy	0.5℃

4. DFRobot *Ambient Light Sensor* (*Brightness Level Sensor*)



Gambar 3.17 DFRobot *Ambient Light Sensor*

Tabel 3.12 Spesifikasi DFRobot *Ambient Light Sensor* [9]

Operating Voltage	2.7 – 6V
Operating Temperature Range	-40℃ - 85℃
Detection Range	0 – 200 klx
Accuracy	0.054 lux

Perbandingan akurasi sensor yang dipakai pada *prototype* dan akurasi sensor yang digunakan untuk perbandingan dapat dilihat ditabel 3.2.2.12.

Tabel 3.13 Tabel Perbandingan Akurasi Sensor

Sensor Suhu & Kelembaban		
Nama	Akurasi Suhu	Akurasi Kelembaban
DFRobot SHT20 Temperature & Humidity Sensor	±0.3 ℃	±3% RH
DHT22	0.5	1%
HDC1080 SHT20	0.2	0.2
DS18B20	0.5	-
Sensor Intensitas Cahaya		
Nama	Akurasi Deteksi Cahaya	

Gravity I2C IP68 Waterproof Ambient Light Sensor	1.2 lux
DFRobot Ambient Light Sensor	0.054 lux

Sebelum melakukan validasi data sensor, ada beberapa library yang penulis pakai untuk dapat menggunakan atau mendapatkan data dari sebuah sensor. Berikut beberapa library yang penulis gunakan untuk melakukan validasi data sensor:

1. `#include <Arduino.h>`

   Library ini adalah *library* dasar yang digunakan untuk melakukan pemograman dengan mikrokontroller Arduino. Contoh seperti: Serial, Delay, Millis dan lain-lain.

2. `#include <Wire.h>`

   Library ini digunakan agar mikrokontroller bisa berkomunikasi dengan sensor yang menggunakan I2C sebagai jalur komunikasi untuk mengirimkan data sensor ke mikrokontoller.

3. `#include <DFRobot_SHT20.h>`

   Library ini digunakan agar ESP32 dapat berkomunikasi dengan sensor DFRobot SHT20, library ini berasal dari tim *developer* DFRobot.

4. `#include <DHT.h>`

   Library ini digunakan untuk melakukan komunikasi antar sensor DHT22 dengan ESP32.

5. `#include <OneWire.h>`

   Library ini digunakan agar ESP32 bisa menerapkan komunikasi One Wire protokol dimana perangkat yang dipakai disambungkan ke satu kabel tunggal. Protokol One Wire ini nantinya akan dipakai pada DS18B20 *Temperature* Sensor. [10]

6. `#include <DallasTemperature.h>`

Library ini digunakan untuk melakukan komunikasi antar sensor DS18B20 dengan ESP32.

7. `#include <ClosedCube_HDC1080.h>`

   Library ini digunakan untuk melakukan komunikasi antar sensor DHC1080 SHT20 dengan ESP32. [11]

8. `#include <DFRobot_B_LUX_V30B.h>`

   Library ini digunakan untuk melakukan komunikasi antar sensor DFRobot Ambient Sensor dengan ESP32, *library* ini berasal dari tim *developer* DFRobot.

Setelah mengumpulkan library yang diperlukan, penulis mulai membandingkan data yang dihasilkan sensor – sensor yang digunakan. Pada percobaan perbandingan pertama, penulis membandingkan sensor suhu dan kelembaban SHT20 dengan DHT22 pada kondisi *indoor* pada siang hari tanpa menggunakan AC. Penulis menempatkan sensor dengan posisi bersebelahan seperti yang terlihat pada gambar 3.2.13 dan setelah beberapa saat data sensor yang muncul pada dilihat pada tabel 3.2.12.

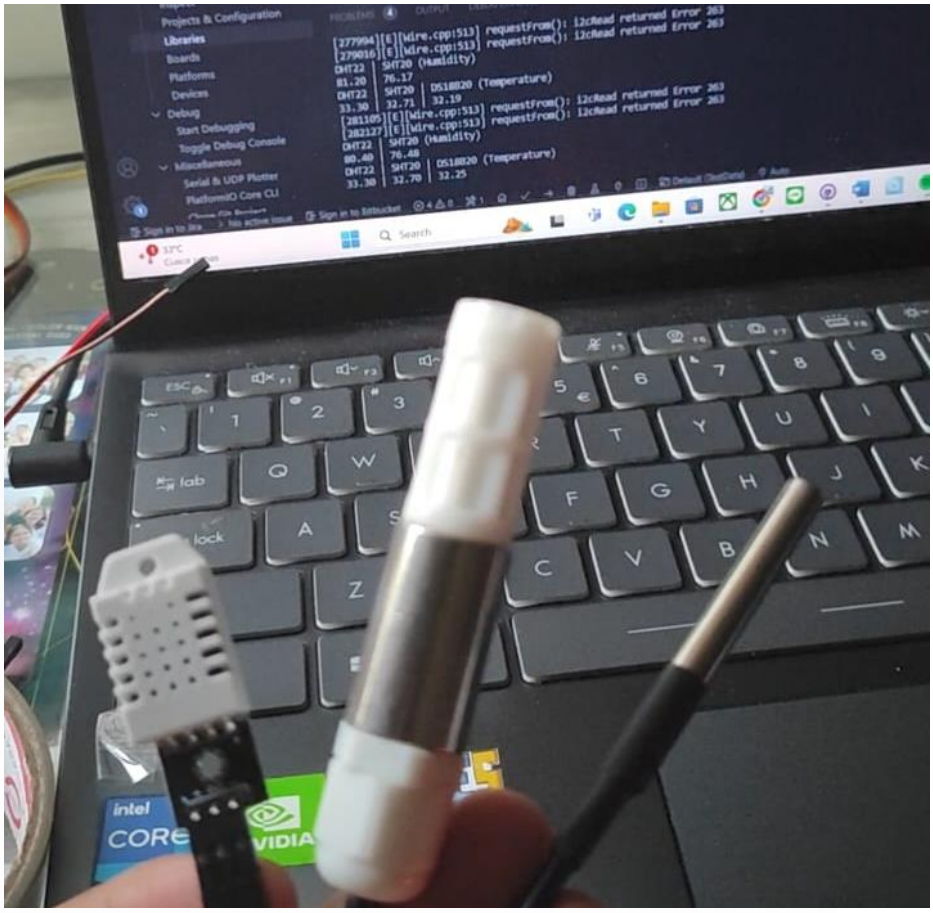
Gambar 3.18 Percobaan Pertama

Tabel 3.14 Data Hasil Percobaan Pertama

Sensor	Data Sensor	
	Temperature	*Humidity*
SHT20	32℃ - 32.7℃	75.7% - 77.5%
DHT22	31.5℃ - 32.5℃	78.2% - 80.5%

Setelah melakukan perbandingan antara data sensor dari SHT20 dan DHT22, terlihat perbedaan yang tidak signifikan yang merupakan hal yang wajar pada sebuah sensor. Hal ini disebabkan oleh akurasi sensor pada kedua sensor dimana SHT20 memiliki akurasi sekitar ±0.3 °C dan ±3% dan DHT22 memiliki akurasi sekitar ±0.5 °C dan ±2%. Perbedaan pada data hasil sensor tersebut masih termasuk di range akurasi pada kedua sensor.

Pada percobaan kedua, penulis menggunakan 3 sensor yaitu SHT20, DHT22 dan DS18B20 untuk membandingkan suhu dan kelembaban pada ruangan indoor tanpa menggunakan AC. Penulis menempatkan ketiga sensor bersebelahan seperti pada gambar 3.2.14 agar mencegah adanya perbedaan kondisi. Hasil data yang didapatkan dapat dilihat di tabel 3.2.13.



Gambar 3.19 Percobaan Kedua

Tabel 3.15 Data Hasil Percobaan Kedua

Sensor	Data Sensor	
	Temperature	*Humidity*
SHT20	32.50℃ - 32.80℃	75.7% - 77.5%
DHT22	32.80℃ - 33.30℃	78.2% - 81.4%
DS18B20	32.20℃ – 32.5℃	-

Setelah melakukan percobaan kedua, penulis mendapatkan bahwa masih belum ada perbedaan yang signifikan pada data dari sensor yang dihasilkan. Hal ini dapat dikatakan bagus karena data yang dihasilkan oleh sensor mencatat sekitar suhu ruangan 32℃ - 33℃ dan kelembapan

pada ruangan sekitar 75 – 81% yang mana perbedaan pada setiap sensor masih dalam range akurasi masing - masing sensor.

Pada percobaan ketiga, penulis menggunakan 2 sensor yaitu DFRobot *Ambient Light Sensor* dan DFRobot *Gravity Light Sensor*. Percobaan dilakukan di ruangan *indoor* yang hanya memiliki sedikit sumber cahaya yang masuk. Penulis tidak bisa melakukan pengujian dengan menggabungkan kedua sensor pada ESP32 karena I2C *address* pada kedua sensor sama (0x40) dan akan bertabrakan satu sama lain. Namun untuk penempatan sensor pada saat pengujian, penulis menggunakan titik yang sama pada meja seperti yang dapat terlihat pada gambar 3.2.15 dan gambar 3.2.16. Setelah melakukan percobaan beberapa saat, data sensor menunjukkan nilai yang berbeda yang tertera pada tabel 3.2.14.

Gambar 3.20 Percobaan Ketiga DFRobot *Ambient Light Sensor*

Gambar 3.21 Percobaan Ketiga DFRobot Gravity I2C IP68 *Waterproof Ambient Light Sensor*

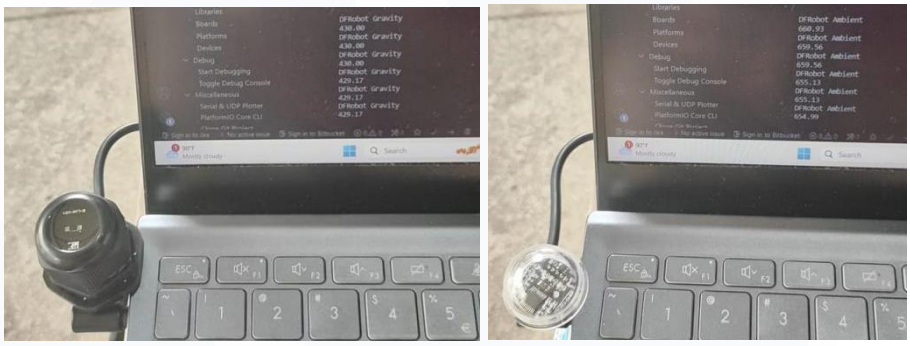
Tabel 3.16 Data Hasil Percobaan Ketiga

Sensor	Data Sensor
DFRobot Gravity	35 – 36 lux
DFRobot Ambient	59 – 60 lux

Setelah melakukan percobaan ketiga, penulis mendapatkan bahwa ada perbedaan data yang cukup signifikan dari yang dihasilkan oleh sensor DFRobot Gravity dan DFRobot Ambient. Terdapat perbedaan sekitar 23 – 25 lux yang mana nilai tersebut melebihi nilai toleransi akurasi dari kedua buah sensor yaitu 0.054 lux untuk DFRobot Ambient Light Sensor dan 1.2 lux untuk DFRobot Gravity IP68 Light Sensor. Maka dapat disimpulkan bahwa pada kondisi indoor, kedua sensor belum bisa menangkap hasil data yang sama.

Pada percobaan ke-empat, penulis menggunakan sensor DFRobot *Ambient Light Sensor* dan DFRobot *Gravity Light Sensor*. Percobaan ini dilakukan di luar ruangan dengan 2 kondisi yaitu sensor tidak berada di bawah langit secara langsung seperti yang terlihat pada gambar 3.2.17 dan gambar 3.2.18 dan sensor berada di bawah langit secara langsung yang tertera pada gambar 3.2.19. Hasil data yang dihasilkan pada

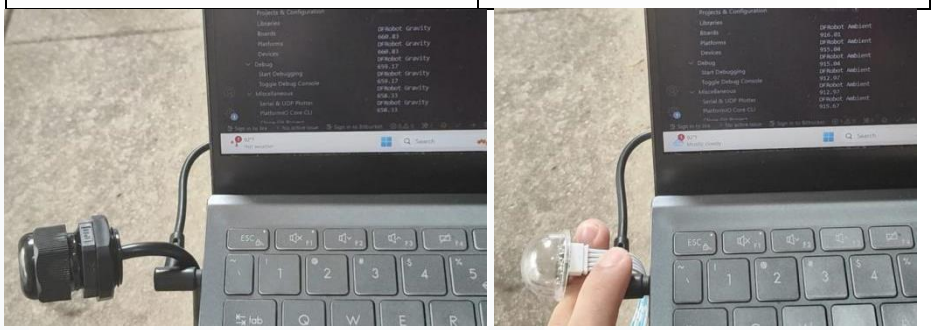
percobaan pada kondisi tidak berada di bawah matahari dapat terlihat pada tabel 3.2.15 dan tabel 3.2.16 dan pada kondisi berada di bawah matahari dapat terlihat pada tabel 3.2.17.



Gambar 3.22 Percobaan keempat pada kondisi tidak berada di bawah langit dengan sensor menghadap ke atas.

Tabel 3.17 Data Hasil Percobaan Pertama keempat pada kondisi tidak berada di bawah langit dengan sensor menghadap ke atas.

Sensor	Data Sensor
DFRobot *Gravity*	420 – 440 lux
DFRobot *Ambient*	650 - 680 lux



Gambar 3.23 Percobaan keempat pada kondisi tidak berada di bawah langit dengan sensor menghadap ke arah sumber cahaya.

Tabel 3.18 Data Hasil Percobaan keempat pada kondisi tidak berada di bawah langit dengan sensor menghadap ke arah sumber cahaya.

Sensor	Data Sensor
DFRobot *Gravity*	640 - 660 lux
DFRobot *Ambient*	900 - 940 lux

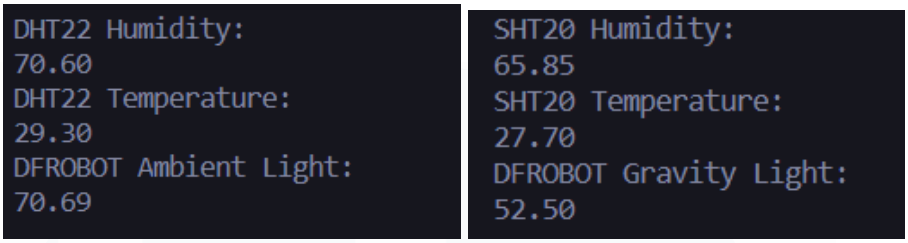
Gambar 3.24 Percobaan keempat pada kondisi berada di bawah langit dengan sensor menghadap ke atas.

Tabel 3.19 Data Hasil Percobaan keempat pada kondisi berada di bawah langit dengan sensor menghadap ke atas.

Sensor	Data Sensor
DFRobot *Gravity*	2290 - 2330 lux
DFRobot *Ambient*	3070 - 3200 lux

Setelah beberapa kali melakukan percobaan dengan kondisi di luar ruangan, kedua sensor intensitas cahaya yang digunakan masih menunjukkan perbedaan angka yang signifikan. Pada sensor DFRobot Gravity IP68 Light Sensor hanya menangkap nilai lux sekitar 62% – 65% dari yang dihasilkan oleh DFRobot Ambient Light Sensor. Maka pada percobaan sensor intensitas cahaya di luar ruangan juga belum bisa menangkap hasil data yang sama.

Pada percobaan kelima, penulis menggunakan 2 ESP32 dengan masing – masing sensor yang berbeda. ESP32 yang pertama menggunakan sensor DHT22 dan DFRobot *Ambient Light Sensor*, sedangkan ESP32 yang kedua menggunakan sensor SHT20 dan DFRobot *Gravity Light Sensor*. Percobaan ini dilakukan di dalam ruangan dengan menggunakan AC dan dengan kondisi minim sumber cahaya. Hasil data yang dihasilkan dari kedua ESP32 dapat dilihat pada gambar 3.2.18.



Gambar 3.25 Hasil data sensor pada percobaan kelima

Lalu beberapa pengujian dan pengembangan yang telah pengulis lakukan, dapat terlihat dari data yang dihasilkan dari sensor terkadang masih menunjukkan perbedaan nilai yang cukup signifikan perbedaannya. Hal ini mungkin bisa disebabkan oleh beberapa hal yaitu:

1. Adanya perbedaaan akurasi, tipe sensor ataupun jalur komunikasi dari sensor ke ESP32 yang berbeda – beda setiap sensor yang menyebabkan setiap sensor dapat menghasilkan data yang berbeda – beda pula.
2. Adanya gangguan pada saat pengiriman data dari sensor ke ESP32, hal ini bisa disebabkan oleh kabel jumper yang kurang baik ataupun *breadboard* yang terkadang tidak tersambung.
3. Perubahan kondisi lingkungan sekitar juga dapat mempengaruhi data yang dihasilkan dari sensor karena setiap sensor memiliki cara masing – masing untuk mengambil dan mengolah data.

Dikarenakan sensor intensitas cahaya yang belum tervalidasi apakah nilai data yang dihasilkan akurat, maka jalan alternatif yang akan digunakan oleh penulis adalah dengan melakukan pengukuran pada lokasi secara langsung untuk mengetahui nilai threshold pada kondisi-kondisi cuaca yang terjadi pada kawasan pertanian salak.

3.3 Kendala yang Ditemukan

Selama proses pengembangan *prototype* pada *project* EPICS IEEE tentunya ada beberapa kendala yaitu:

1. Dalam pengembangan *prototype*, pemilihan komponen menjadi salah satu kendala karena sensor yang dipilih harus memiliki spesifikasi yang sesuai dengan yang keperluan *project* dimana *prototype* akan digunakan di luar ruangan.
2. Sensor yang digunakan terlalu banyak sehingga ada beberapa sensor yang memiliki kendala seperti *address* I2C yang bertabrakan.
3. Pada saat pengembangan *prototype*, terdapat beberapa *hardware* yang rusak / tidak berfungsi.
4. Data sensor yang dihasilkan oleh sensor yang digunakan belum tervalidasi apakah data tersebut akurat atau tidak.

3.4 Solusi atas Kendala yang Ditemukan

Berikut solusi yang penulis pakai untuk menghadapi kendala yang ditemukan pada saat melakukan kerja magang:

1. Penulis membuat daftar hardware yang kira – kira cocok untuk dipakai pada *prototype*, lalu penulis beserta tim lain berpikir dan menentukan sensor mana yang cocok untuk digunakan pada *prototype* dengan mempertimbangkan spesifikasi dan kebutuhan.
2. Penulis mencari alternatif sensor lain yang sekiranya dapat digunakan pada *prototype* agar mencegah terjadinya I2C address yang bertabrakan.
3. Penulis selanjutnya membeli peralatan / *hardware* cadangan agar jika terjadi sesuatu pada saat pengembangan, ada *hardware* cadangan yang siap digunakan.
4. Penulis membandingkan data yang berasal dari beberapa sensor yang memiliki spesifikasi dan akurasi yang mendekati sensor yang akan digunakan untuk mengetahui apakah hasil data sensor yang dipakai akurat.