JOB 1 Instalasi ESP32 pada Arduino IDE

I. Tujuan

Peserta didik dapat mengetahui dan mengerti tata cara instalasi board ESP32 pada *software* Arduino IDE.

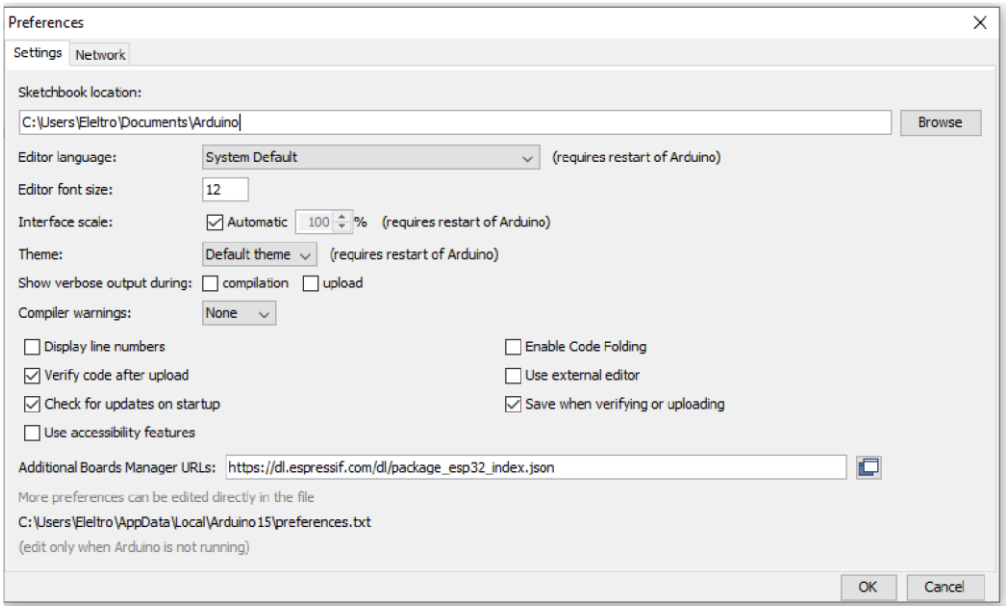
II. Alat dan Bahan

Adapun yang harus disediakan yaitu:

1. Komputer/Laptop dengan koneksi internet.
2. *Software* Arduino IDE.

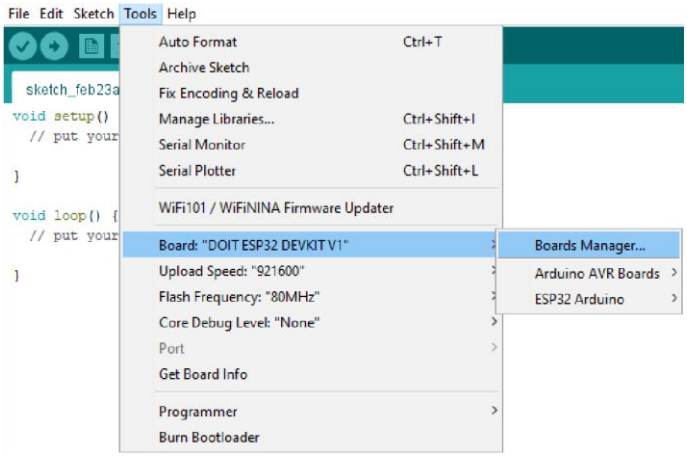
III. Langkah Kerja

1. Buka *software* Arduino IDE pada komputer/laptop, klik menu **File > Preferences** atau bisa dengan tekan **Ctrl+Comma** sehingga muncul jendela seperti ini.

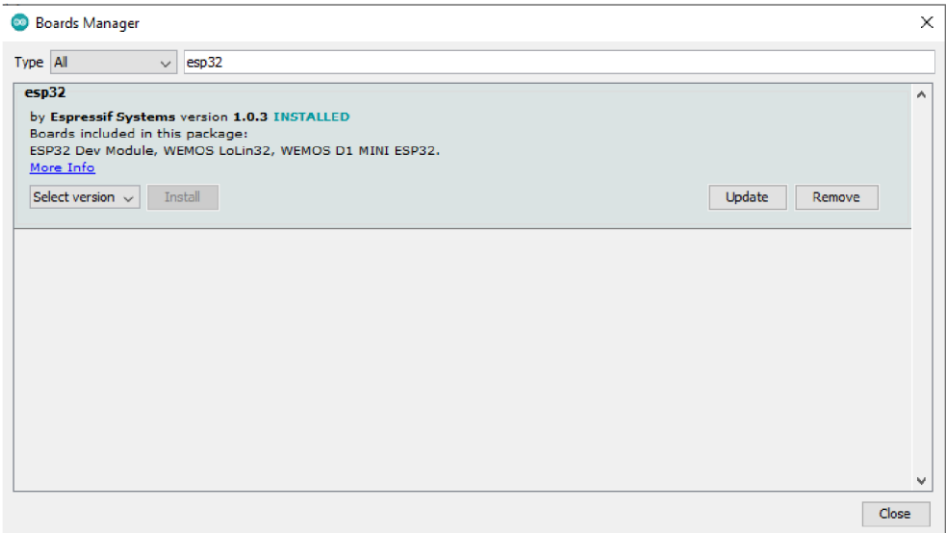


2. Terdapat kolom **Additional Board Manager URLs**, lalu isi kolom dengan mengetik *link* https://dl.espressif.com/dl/package_esp32_index.json lalu klik **OK.**

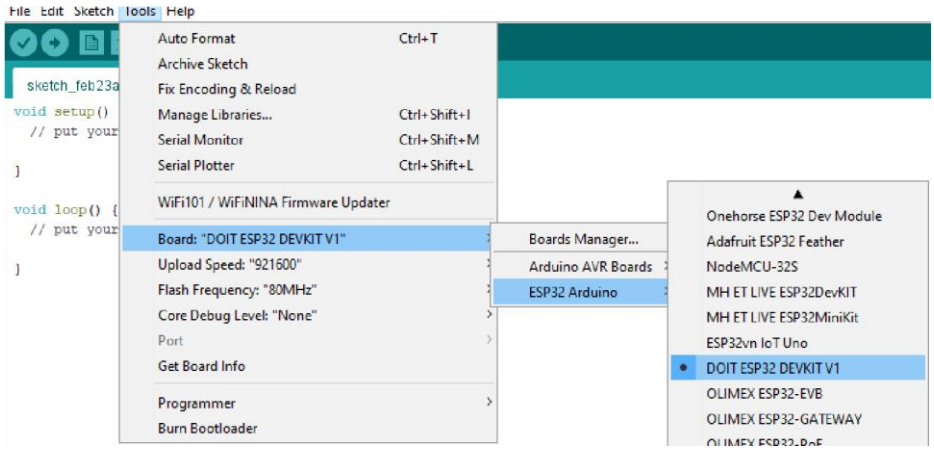
3. Tutup jendela **Preferences** kemudian klik menu **Tools > Boards > Boards manager**.



4. Pastikan komputer/laptop terhubung dengan koneksi internet. Pada jendela **board Manager** ketik ESP32 pada kolom pencarian.



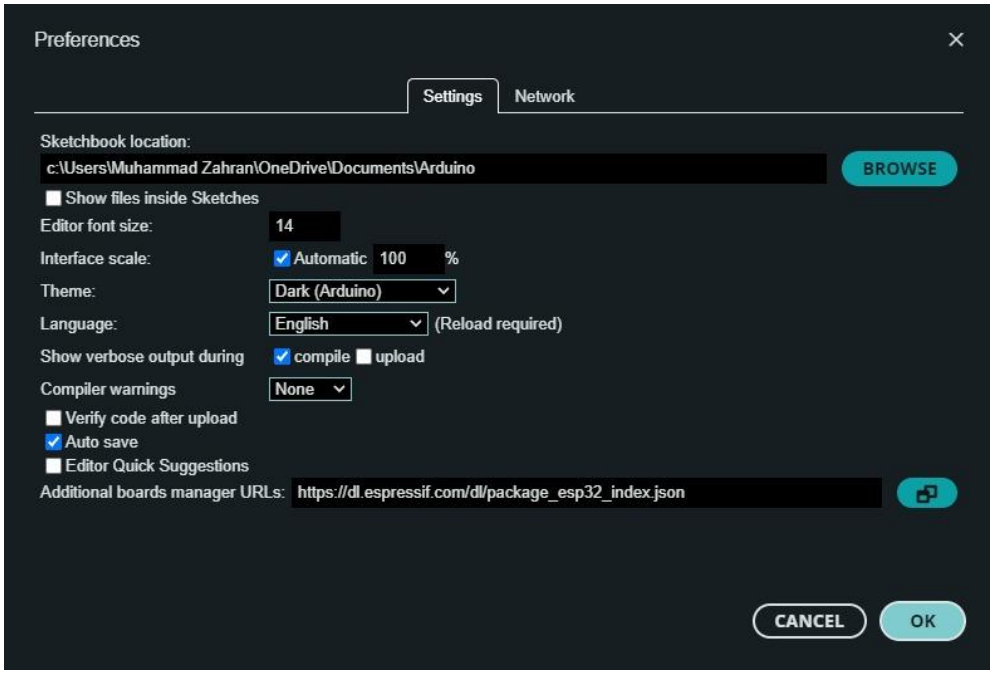
5. Klik **Install** dan tunggu beberapa saat sampai proses instalasi selesai lalu klik **close**. Proses instalasi board ESP32 pada Arduino IDE telah selesai dan siap untuk digunakan.
6. Agar dapat mengupload hasil pembuatan program ke board ESP32 maka harus dilakukan *setting* terlebih dahulu. Tahapannya klik **Tools > Board > ESP32 Arduino >** lalu pilih board ESP32 sesuai dengan seri yang digunakan. Pada trainer ini board ESP32 yang digunakan yaitu **DOIT ESP32 DevkitVI**.

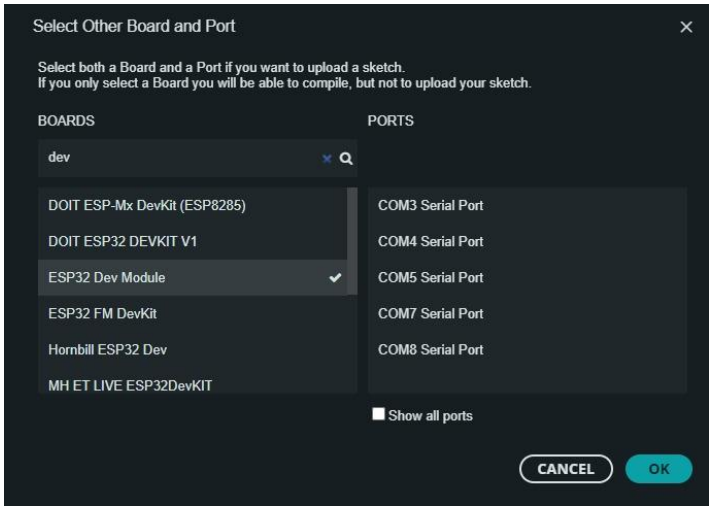


7. Jika tahap 1-6 telah dilakukan maka di setiap pembuatan program selanjutnya tahap ini tidak perlu diulangi.
8. Setelah proses *setting* selesai maka pengguna dapat mulai membuat program yang kemudian dapat diupload dengan menyesuaikan dengan *port* yang telah terhubung dengan ESP32 terlebih dahulu.

IV. Hasil Percobaan

1. Install board ESP32





V. Kesimpulan

Kesimpulan dari Praktikum Job 1 ini adalah board untuk menggunakan ESP32 tidak otomatis tersedia dalam Arduino IDE. Jadi harus menginstall board manager eksternal yang bisa di akses melalui link diatas.

JOB 2 Memprogram Sensor pada ESP 32

I. Tujuan

Peserta didik dapat memprogram berbagai sensor menggunakan ESP32 serta dapat mengimplementasikan dalam pembuatan project.

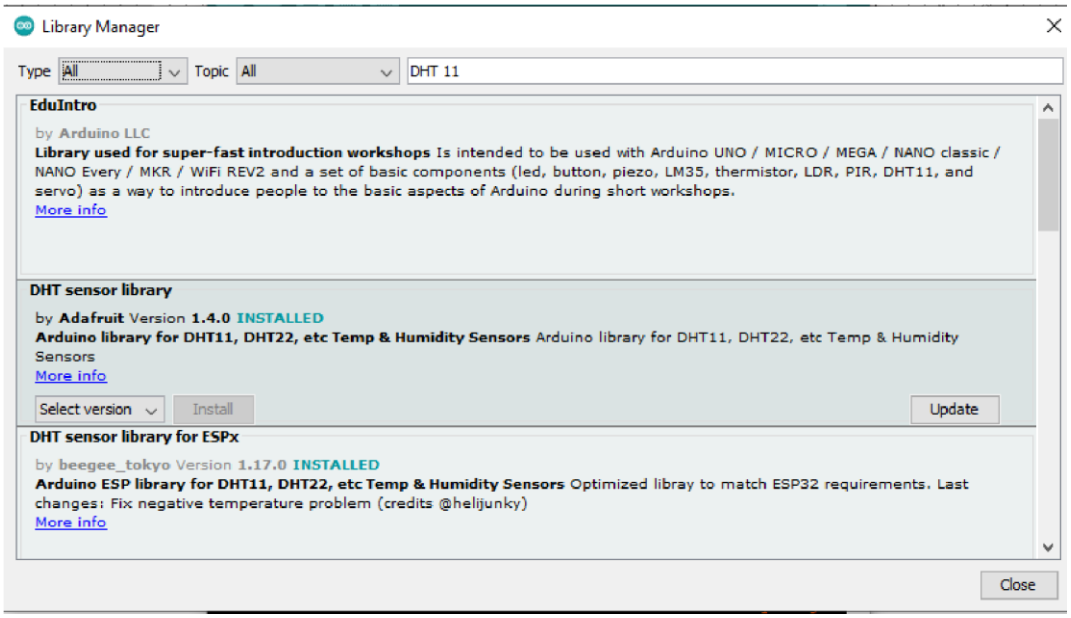
II. Alat dan Bahan

Adapun yang harus disediakan yaitu:

1. Komputer/Laptop dengan koneksi internet.
2. *Software* Arduino IDE.
3. Aplikasi TeamViewer

III. Langkah Kerja

1. Jika praktikum dilaksanakan secara *offline*, buka *software* Arduino IDE yang ada pada komputer anda dan mulai memprogram.
2. Beberapa sensor yang digunakan memerlukan *library* dari Arduino IDE, pastikan pada *software* anda telah terpasang *library* yang dibutuhkan. Sensor yang memerlukan *library* yaitu DHT11 menggunakan DHT11 sensor library dan Adafruit unified sensor.
3. Untuk menambahkan library klik **Sketch > Include Library** lalu pilih library yang akan digunakan, jika belum tersedia harus install terlebih dahulu di Library Manager dengan cara klik **Sketch > Include Library > Manage Libraries** dan cari library yang akan digunakan lalu install seperti contoh di bawah ini



4. Jika praktikum dilaksanakan secara *online* maka pastikan laptop atau komputer anda terhubung dengan koneksi internet lalu ikuti teknis pelaksanaan praktikum.
5. Lakukan pemrograman dengan menjalankan satu per satu pada masing masing sensor yang ada pada trainer.
6. *Compile* program sampai tidak ada *error*
7. *Upload* program sesuaikan dengan port yang digunakan pastikan sesuai dengan node yang dituju.
8. Setelah berhasil *upload* program lihat hasil melalui serial monitor, jika pada serial monitor tidak menunjukan hasil maka ganti *baud rate* pada serial monitor sesuaikan dengan yang ada pada program lalu *reset* trainer**.**

IV. Program

Program	Coding
Node 1	

DHT 11	`#include <Adafruit_Sensor.h> #include <DHT.h>`
	
	`#define DHTPIN 4`

	#define DHTTYPE DHT11
	DHT dht(DHTPIN, DHTTYPE);
	
	void setup() { Serial.begin(115200);
	Serial.println(F("DHTxx test!"));
	dht.begin();
	} void
	loop() {
	// Reading temperature or humidity takes about 250 milliseconds!
	// Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor) float h = dht.readHumidity(); // Read temperature as Celsius (the default) float t = dht.readTemperature();
	// Read temperature as Fahrenheit (isFahrenheit = true) float f = dht.readTemperature(true);
	// Check if any reads failed and exit early (to try again).
	
	if (isnan(h) || isnan(t) || isnan(f)) { Serial.println(F("Failed to read from DHT sensor!")); return;
	}
	Serial.print(F("Humidity: "));
	Serial.print(h);
	Serial.print(F("% Temperature: "));
	Serial.print(t);
	Serial.print(F("°C "));
	}

Passive Infrared	int pin = 27;

	void setup()
	{
	pinMode(pin,INPUT);
	Serial.begin(115200);
	} void
	loop()
	{
	bool isDetected = digitalRead(pin);
	if(isDetected){
	Serial.println("Motion detected");
	}
	delay(500);
	}

LDR	int ldr = 13;
	int nilai;
	void setup() {
	Serial.begin(115200);
	Serial.print("LDR TEST");
	} void
	loop() {
	nilai = analogRead(ldr);
	Serial.print("Nilai LDR: ");
	Serial.println(nilai);
	}

Node 2	

LM35	int lm35 = 27; float
	suhu = 00; int
	suhu1=00;
	
	void setup(){
	Serial.begin(115200);
	}
	void loop(){ suhu1 =
	analogRead(lm35);

	suhu = suhu1 / 2.0479;
	Serial.println(suhu);
	delay(50);
	}

Infrared	int sensor = 13;
	void setup(){
	Serial.begin(115200);
	} void
	loop() {
	
	int hasil = digitalRead(sensor);
	if (hasil == LOW)
	{
	Serial.println("Hambatan Terdeteksi");
	} if (hasil ==
	HIGH)
	{
	Serial.println ("Tidak Ada Hambatan");
	}
	delay(200);
	}

	int tilt = 4;

Tilt	void setup()
	{
	Serial.prin
	} void 115200);
	loop()
	{
	
	Serial.prin
	if(nilaiTi lt = digitalRead(tlit);
	HIGH) nilaiTilt);
	{
	
	}
	intln("Miring");

	else
	{
	rial.print("Tegak");
	y(200);
	}
	
	}

Node 3	

Touch	const int SENSOR_PIN = 13;
	int lastState = LOW;
	int currentState;
	
	void setup() {
	  Serial.begin(115200);
	pinMode(SENSOR_PIN, INPUT);
	}
	void loop() {
	  currentState = digitalRead(SENSOR_PIN);
	
	  if(lastState == LOW && currentState == HIGH)
	Serial.println("The sensor is touched"); lastState
	= currentState;
	}

Sound Sensor	int sound = 27;
	
	void setup() {
	Serial.begin(115200);
	} void loop() { int baca_sensor =
	digitalRead(sound); if (baca_sensor
	== 1) { Serial.println("ada
	suara");
	  }
	else {
	  Serial.println("suara mati");
	  }

	delay(500); }

Vibration Sensor	int vib = 4;
	
	void setup(){
	Serial.begin(115200);
	}
	void loop(){
	Int value = pulseIn (vib, HIGH); delay(50);
	Serial.println(value); if (value >
	1000){ Serial.println("Vibration
	detection");
	}
	}

V. Tugas

Buatlah program untuk menggabungkan masing-masing sensor yang berada pada node yang sama dengan cara

1. Lakukan pemrograman dengan menggabungkan beberapa sensor yang ada pada masing-masing node. Untuk mempermudah melakukan penggabungan beberapa jenis program menjadi satu program ada berbagai macam cara yang bisa dilakukan dan pada jobsheet ini penulis akan memberikan salah satu metode yang mudah untuk ditiru, adapun langkah langkahnya sebagai berikut :
   a. Buat program baru dan buka program – program yang akan digabungkan menjadi satu program.
   b. Gabungkan seluruh header dari berbagai program yang berbeda dan jika ada yang sama fungsinya maka cukup tulis satu saja. Contoh :

      ```
      #include library 1
      #include library 2
      ...
      #include library – n
      ```

   c. Gabungkan seluruh deklarasi variabel dari berbagai program yang berbeda dan jika ada yang sama fungsinya maka cukup tulis satu saja.

Contoh :

```
deklarasi variabel 1
deklarasi variabel 2
...
deklarasi variabel -n
```

d. Jangan langsung isi void setup( ) dengan seluruh baris program dari void setup( ) masing – masing program yang yang akan digabungkan. Baris paling awal pada void setup () yang harus ditulis adalah cukup Serial.begin(115200) pastikan hanya ada satu Serial.begin(115200) pada Void setup( ) Contoh

:

```
Void setup(){
 Serial.begin(115200);
.....
   }
```

Setelah itu, buat fungsi void yang dapat dipanggil (Callback Function) untuk menyimpan masing – masing isi dari void setup program yang yang akan digabungkan, lalu header dari fungsi tersebut masukan ke dalam Void setup( ) utama. Contoh 2 void setup yang akan digabungkan dari program **DHT 11** dan **Sensor LDR** :

1) **DHT 11**

```
   void setup( ) {
   Serial.println(F("DHTxx test!"));
dht.begin(); }
```

2) **Sensor LDR**

```
   void setup() {
   Serial.begin(115200);
   Serial.print("LDR TEST");
       }
```

Digabungkan menjadi :

```
   void setup( ) {
```

```
Serial.begin(115200);
setupDHT ()
setupLDR( )  } void
loop() {
  .....  } void
setup11( ){
Serial.println(F("DHTxx test!"));
dht.begin();
}
void setupIRO( ){ Serial.print("LDR
TEST");
}
```

e. Untuk void loop() dapat berlaku hal yang sama seperti void setup (), namun tidak selamanya hal ini bisa digunakan pada void loop(), bergantung pada program yang akan dibuat. Sehingga pada void loop() umumnya dilakukan penggabungan langsung dengan penyesuaian yang dibutuhkan.

2. Compile program sampai tidak ada error.
3. Upload program sesuaikan dengan port yang digunakan pastikan sesuai dengan node yang dituju.
4. Setelah berhasil upload program lihat hasil melalui serial monitor, jika pada serial monitor tidak menunjukan hasil maka ganti baud rate pada serial monitor sesuaikan dengan yang ada pada program lalu reset trainer**.**

VI. Hasil Praktikum

1. Program DHT11 Sensor

```
#include <DHT11.h>

// Create an instance of the DHT11 class and set the digital I/O pin.
DHT11 dht11(32);

void setup()
{
    // Initialize serial communication at 115200 baud.
```

```
    Serial.begin(115200);
}

void loop()
{
    // Read the humidity from the sensor.
    float humidity = dht11.readHumidity();

    // Read the temperature from the sensor.
    float temperature = dht11.readTemperature();

    // If the temperature and humidity readings were successful, print them to
the serial monitor.
    if (temperature != -1 && humidity != -1)
    {
        Serial.print("Temperature: ");
        Serial.print(temperature);
        Serial.println(" C");

        Serial.print("Humidity: ");
        Serial.print(humidity);
        Serial.println(" %");
    }
    else
    {
        // If the temperature or humidity reading failed, print an error
message.
        Serial.println("Error reading data");
    }

    // Wait for 2 seconds before the next reading.
    delay(2000);
}
```

Hasil program

2. Program MQ-2

```
 #define mq2Pin 32   // Pin analog untuk sensor MQ-2
#include <Wire.h>

void setup() {
  Serial.begin(9600);
  pinMode(mq2Pin , INPUT);
}

void loop() {
  int sensorValue = analogRead(mq2Pin); // Membaca nilai sensor analog

  float voltage = sensorValue * (5.0 / 1023.0); // Mengonversi nilai sensor menjadi tegangan (5V adalah tegangan referensi Arduino)

  // Menghitung konsentrasi gas menggunakan rumus yang sesuai dengan sensor MQ-2
  float gasResistance = ((5.0 - voltage) / voltage) * 10.0; // Menggunakan faktor 10.0 untuk mengkoreksi nilai

  // Menampilkan hasil ke Serial Monitor
  Serial.print("Sensor Value: ");
  Serial.println(sensorValue);
  delay(2000);
  Serial.print("Gas Resistance: ");
  Serial.print(gasResistance);
  Serial.println(" KΩ");
  delay(2000);

}
```

Hasil Program

3. Program DHT11

```
int SensorPin = 32;// deklarasi pin analog yg dipakai
int soilMoistureValue; // menyimpan nilai analog dari sensor ke esp32
int soilmoisturepercent; // nilai yg diperoleh dalam bentuk persen setelah
dimaping

void setup() {
  Serial.begin(115200); // Baudrate komunikasi dengan serial monitor
}

void loop() {
      soilMoistureValue = analogRead(SensorPin);
        Serial.print("Nilai analog = ");
          Serial.print(soilMoistureValue);
           soilmoisturepercent = map(soilMoistureValue, 4095, 0, 0, 100);

        Serial.print(" Presentase kelembaban tanah= ");
       Serial.print(soilmoisturepercent);
      Serial.println("% ");


delay(500);
}
```

Hasil Program

VII. Kesimpulan

Setelah melakukan percobaan, kami berkesimpulan bahwa membuat program sensor untuk ESP32 hampir sama dengan membuat program dengan Arduino.

JOB 3 Topologi *Wi-fi Mesh Network*

I. Tujuan

Peserta didik dapat memprogram ESP32 dengan cara menghubungkan antar *board* menggunakan topologi Wi-Fi Mesh Network

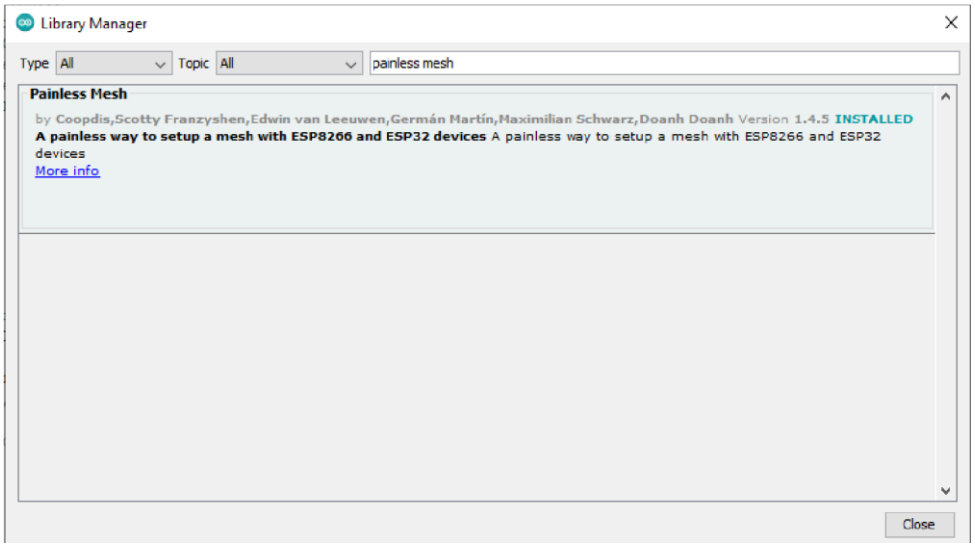
II. Alat dan Bahan

Adapun yang harus disediakan yaitu:

1. Komputer/Laptop dengan koneksi internet.
2. *Software* Arduino IDE.
3. Aplikasi TeamViewer

III. Langkah Kerja

1. Jika praktikum dilaksanakan secara *offline*, buka *software* Arduino IDE yang ada pada komputer anda dan mulai memprogram.
2. Topilogi wifi mesh network ini memerlukan *library* dari Arduino IDE, pastikan pada *software* anda telah terpasang *library* yang dibutuhkan, jika *library* belum terpasang pada Arduino IDE klik **Sketch > Include Library > Manage Libraries** lalu ketik pada kolom pencarian “painless mesh” dan install



3. Jika praktikum dilaksanakan secara *online* maka pastikan laptop atau komputer anda terhubung dengan koneksi internet lalu ikuti teknis pelaksanaan praktikum.
4. Lakukan pemrograman dengan menjalankan ketiga node. Setelah program berhasil di*compile* lalu *upload* program sesuaikan dengan port yang digunakan pastikan sesuai dengan node yang dituju.
5. Setelah berhasil *upload* program lihat hasil melalui serial monitor, jika pada serial monitor tidak menunjukan hasil maka ganti *baud rate* pada serial monitor sesuaikan dengan yang ada pada program lalu *reset* trainer**.**

IV. Program

Program topologi wifi mesh network

```
#include <painlessMesh.h>
#define   LED             2
#define   BLINK_PERIOD    3000
#define   BLINK_DURATION  100
#define   MESH_SSID       "ESP32 Remote Lab"
#define   MESH_PASSWORD   "1234567890"
#define   MESH_PORT       5555

// Prototypes void
sendMessage();
```

```
void
receivedCallback(u
int32_t from,
String & msg);
void
newConnectionCallb
ack(uint32_t
nodeId); void
changedConnectionC
allback();  void
nodeTimeAdjustedCa
llback(int32_t
offset);  void
delayReceivedCallb
ack(uint32_t from,
int32_t delay);

Scheduler     userScheduler; // to control your personal task
painlessMesh        mesh;    bool    calc_delay    =    false;
SimpleList<uint32_t> nodes;

void sendMessage() ; // Prototype
Task taskSendMessage( TASK_SECOND * 1, TASK_FOREVER,
&sendMessage ); // start with a one second interval
// Task to blink the number of nodes
Task blinkNoNodes; bool onFlag =
false;
 void setup() {   Serial.begin(115200);   pinMode(LED,
OUTPUT);   mesh.setDebugMsgTypes(ERROR | DEBUG);  // set
before init() so that you can see error messages
  mesh.init(MESH_SSID, MESH_PASSWORD, &userScheduler,
MESH_PORT);   mesh.onReceive(&receivedCallback);
mesh.onNewConnection(&newConnectionCallback);
mesh.onChangedConnections(&changedConnectionCallback);
mesh.onNodeTimeAdjusted(&nodeTimeAdjustedCallback);
mesh.onNodeDelayReceived(&delayReceivedCallback);
```

```
userScheduler.addTask( taskSendMessage );
taskSendMessage.enable();
   blinkNoNodes.set(BLINK_PERIOD, (mesh.getNodeList().size()
+
1) * 2, []() {
      // If on, switch off, else switch on
      if (onFlag)         onFlag = false;       else
onFlag = true;       blinkNoNodes.delay(BLINK_DURATION);
if (blinkNoNodes.isLastIteration()) {         // Finished
blinking. Reset task for next run          // blink number
of nodes (including this node) times
blinkNoNodes.setIterations((mesh.getNodeList().size()
+ 1) * 2);
        // Calculate delay based on current mesh time and
BLINK_PERIOD
        // This results in blinks between nodes being synced
blinkNoNodes.enableDelayed(BLINK_PERIOD -
(mesh.getNodeTime() % (BLINK_PERIOD*1000))/1000);
      }   });
userScheduler.addTask(blinkNoNodes);
blinkNoNodes.enable();
randomSeed(analogRead(A0));
}  void loop() {
mesh.update();
digitalWrite(LED, !onFlag);
}
void sendMessage() {   String msg = "Hello from node
";   msg += mesh.getNodeId();   msg += " myFreeMemory:
" + String(ESP.getFreeHeap());
mesh.sendBroadcast(msg);   if (calc_delay) {
    SimpleList<uint32_t>::iterator node = nodes.begin();
while (node != nodes.end()) {
mesh.startDelayMeas(*node);
      node++;     }
calc_delay = false;
  }
```

```
  Serial.printf("Sending message: %s\n", msg.c_str());
taskSendMessage.setInterval( random(TASK_SECOND * 1,
TASK_SECOND * 5));  // between 1 and 5 seconds
}
void receivedCallback(uint32_t from, String & msg) {
Serial.printf("startHere: Received from %u msg=%s\n", from,
msg.c_str());
}  void newConnectionCallback(uint32_t nodeId) {   // Reset
blink task   onFlag = false;
blinkNoNodes.setIterations((mesh.getNodeList().size() + 1) *
2);   blinkNoNodes.enableDelayed(BLINK_PERIOD -
(mesh.getNodeTime() %
(BLINK_PERIOD*1000))/1000);

  Serial.printf("--> startHere: New Connection, nodeId =
%u\n", nodeId);
  Serial.printf("--> startHere: New Connection, %s\n",
mesh.subConnectionJson(true).c_str());
}
void changedConnectionCallback() {   Serial.printf("Changed
connections\n");   // Reset blink task   onFlag = false;
blinkNoNodes.setIterations((mesh.getNodeList().size() + 1) *
2);
blinkNoNodes.enableDelayed(BLINK_PERIOD -
(mesh.getNodeTime() % (BLINK_PERIOD*1000))/1000);
  nodes = mesh.getNodeList();
Serial.printf("Num nodes: %d\n", nodes.size());
  Serial.printf("Connection list:");
SimpleList<uint32_t>::iterator node = nodes.begin();
while (node != nodes.end()) {     Serial.printf("
%u", *node);     node++;
  }
  Serial.println();
calc_delay = true;
}
```

```
void nodeTimeAdjustedCallback(int32_t offset) {
Serial.printf("Adjusted time %u. Offset = %d\n",
mesh.getNodeTime(), offset);
}
void delayReceivedCallback(uint32_t from, int32_t delay) {
Serial.printf("Delay to node %u is %d us\n", from, delay);
}
```

V. Hasil Percobaan

1. Program

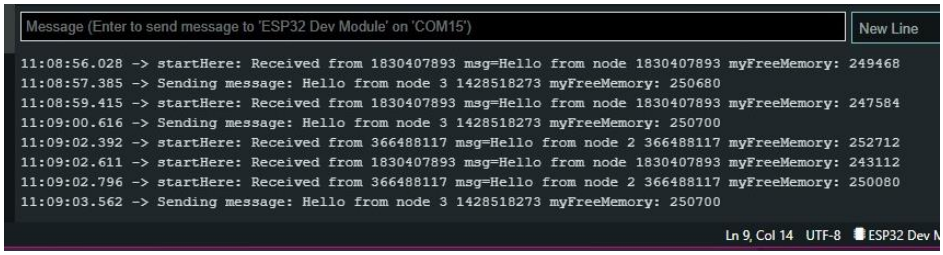
```
#include <painlessMesh.h>
#include <AsyncTCP.h>
#define LED 2
#define BLINK_PERIOD 3000
#define BLINK_DURATION 100
#define MESH_SSID "ESP32 Remote Lab"
#define MESH_PASSWORD "1234567890"
#define MESH_PORT 5555
// Prototypes
void sendMessage();

void receivedCallback(uint32_t from, String & msg);
void newConnectionCallback(uint32_t nodeId);
void changedConnectionCallback();
void nodeTimeAdjustedCallback(int32_t offset);
void delayReceivedCallback(uint32_t from, int32_t delay);
Scheduler userScheduler; // to control your personal task
painlessMesh mesh;
bool calc_delay = false;
SimpleList<uint32_t> nodes;
void sendMessage() ; // Prototype
Task taskSendMessage( TASK_SECOND * 1, TASK_FOREVER,
&sendMessage ); // start with a one second interval
// Task to blink the number of nodes
Task blinkNoNodes;
bool onFlag = false;
void setup() {
 Serial.begin(9600);
 pinMode(LED, OUTPUT);
```

```
 mesh.setDebugMsgTypes(ERROR | DEBUG); // set before init() so that you can
see error messages
 mesh.init(MESH_SSID, MESH_PASSWORD, &userScheduler, MESH_PORT);
 mesh.onReceive(&receivedCallback);
 mesh.onNewConnection(&newConnectionCallback);
 mesh.onChangedConnections(&changedConnectionCallback);
 mesh.onNodeTimeAdjusted(&nodeTimeAdjustedCallback);
 mesh.onNodeDelayReceived(&delayReceivedCallback);
 userScheduler.addTask( taskSendMessage );
 taskSendMessage.enable();
 blinkNoNodes.set(BLINK_PERIOD, (mesh.getNodeList().size() + 1) * 2, []() {
 // If on, switch off, else switch on

 if (onFlag)
 onFlag = false;
 else
 onFlag = true;
 blinkNoNodes.delay(BLINK_DURATION);
 if (blinkNoNodes.isLastIteration()) {
 // Finished blinking. Reset task for next run
 // blink number of nodes (including this node) times
 blinkNoNodes.setIterations((mesh.getNodeList().size() + 1) * 2);
 // Calculate delay based on current mesh time and
 BLINK_PERIOD;
 // This results in blinks between nodes being synced
 blinkNoNodes.enableDelayed (BLINK_PERIOD - (mesh.getNodeTime() %
(BLINK_PERIOD*1000))/1000);
 }
 });
 userScheduler.addTask(blinkNoNodes);
 blinkNoNodes.enable();
 randomSeed(analogRead(A0));
}
void loop() {
 mesh.update();
 digitalWrite(LED, !onFlag);
}
void sendMessage() {
 String msg = "Hello from node ";
 msg += mesh.getNodeId();
 msg += " myFreeMemory: " + String(ESP.getFreeHeap());
 mesh.sendBroadcast(msg);
 if (calc_delay) {
```

```
 SimpleList<uint32_t>::iterator node = nodes.begin();
 while (node != nodes.end()) {
 mesh.startDelayMeas(*node);

 node++;
 }
 calc_delay = false;
 }
 Serial.printf("Sending message: %s\n", msg.c_str());
 taskSendMessage.setInterval( random(TASK_SECOND * 1, TASK_SECOND * 5)); //
between 1 and 5 seconds
}
void receivedCallback(uint32_t from, String & msg) {
 Serial.printf("startHere: Received from %u msg=%s\n", from, msg.c_str());
}
void newConnectionCallback(uint32_t nodeId) {
 // Reset blink task
 onFlag = false;
 blinkNoNodes.setIterations((mesh.getNodeList().size() + 1) * 2);
 blinkNoNodes.enableDelayed(BLINK_PERIOD -
(mesh.getNodeTime() % (BLINK_PERIOD*1000))/1000);
 Serial.printf("--> startHere: New Connection, nodeId = %u\n", nodeId);
 Serial.printf("--> startHere: New Connection, %s\n",
mesh.subConnectionJson(true).c_str());
}
void changedConnectionCallback() {
 Serial.printf("Changed connections\n");
 // Reset blink task
 onFlag = false;
 blinkNoNodes.setIterations((mesh.getNodeList().size() + 1) * 2);
 blinkNoNodes.enableDelayed(BLINK_PERIOD - (mesh.getNodeTime() %
(BLINK_PERIOD*1000))/1000);

 nodes = mesh.getNodeList();
 Serial.printf("Num nodes: %d\n", nodes.size());
 Serial.printf("Connection list:");
 SimpleList<uint32_t>::iterator node = nodes.begin();
 while (node != nodes.end()) {
 Serial.printf(" %u", *node);
 node++;
 }
 Serial.println();
 calc_delay = true;
```

```
}
void nodeTimeAdjustedCallback(int32_t offset) {
 Serial.printf("Adjusted time %u. Offset = %d\n", mesh.getNodeTime(), offset);
}
void delayReceivedCallback(uint32_t from, int32_t delay) {
 Serial.printf("Delay to node %u is %d us\n", from, delay);
}
```

Hasil Program



VI. Kesimpulan

Dengan menyamakan MESH SSID dan MESH PASSWORD akan membuat pertukaran data pada masing-masing Node.

JOB 4 Wi-fi Mesh Wireless Network Sensor

I. Tujuan

Peserta didik dapat memprogram ESP32 dengan bertukar data sensor yang dimiliki antar node menggunakan topologi Wi-Fi Mesh Network

II. Alat dan Bahan

Adapun yang harus disediakan yaitu:

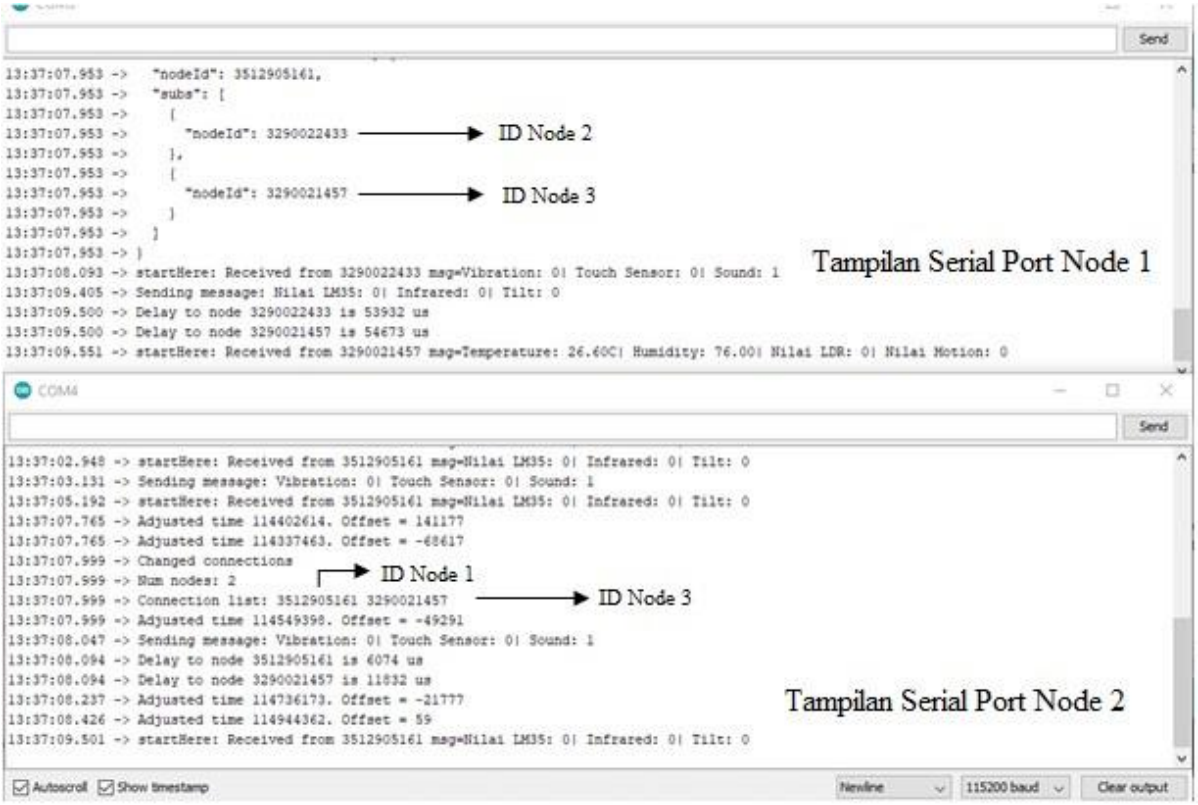
1. Komputer/Laptop dengan koneksi internet.
2. *Software* Arduino IDE.
3. Aplikasi TeamViewer

III. Langkah Kerja

1. Jika praktikum dilaksanakan secara *offline*, buka *software* Arduino IDE yang ada pada komputer anda dan mulai memprogram.
2. Karena menggunakan sensor dan topologi wifi mesh network, pastikan pada Arduino IDE telah terinstall *library* yang akan digunakan dalam memprogram.
3. Jika praktikum dilaksanakan secara *online* maka pastikan laptop atau komputer anda terhubung dengan koneksi internet lalu ikuti teknis pelaksanaan praktikum.
4. Lakukan pemrograman dengan menjalankan ketiga node. Setelah program berhasil di*compile* lalu *upload* program sesuaikan dengan port yang digunakan pastikan sesuai dengan node yang dituju.
5. Untuk bertukar data antar node ubah program bagian ini dengan data yang akan diambil dari sensor

```
String msg = "Hello from node ";   msg += mesh.getNodeId();   msg += " myFreeMemory: " + String(ESP.getFreeHeap());   mesh.sendBroadcast(msg);
```

Bagian ini adalah bagian untuk diubah dengan format untuk mengambil data yang akan ditampilkan dan dikirimkan. Sehingga hasil yang diharapkan yaitu antar node dapat bertukar data dari sensor seperti contoh di bawah ini



IV. Hasil Praktikum

1. Program Node 1: DHT11

```
#include <painlessMesh.h>
#include <AsyncTCP.h>
#include <DHT11.h>
#define LED 2
#define BLINK_PERIOD 3000
#define BLINK_DURATION 100
#define MESH_SSID "ESP32 Remote Lab"
#define MESH_PASSWORD "1234567890"
#define MESH_PORT 5555

DHT11 dht11(32);
```

```
// Prototypes
void sendMessage();

void receivedCallback(uint32_t from, String & msg);
void newConnectionCallback(uint32_t nodeId);
void changedConnectionCallback();
void nodeTimeAdjustedCallback(int32_t offset);
void delayReceivedCallback(uint32_t from, int32_t delay);
Scheduler userScheduler; // to control your personal task
painlessMesh mesh;
bool calc_delay = false;
SimpleList<uint32_t> nodes;
void sendMessage() ; // Prototype
Task taskSendMessage( TASK_SECOND * 1, TASK_FOREVER,
&sendMessage ); // start with a one second interval
// Task to blink the number of nodes
Task blinkNoNodes;
bool onFlag = false;
void setup() {
 Serial.begin(9600);
 pinMode(LED, OUTPUT);
 mesh.setDebugMsgTypes(ERROR | DEBUG); // set before init() so that you can
see error messages
 mesh.init(MESH_SSID, MESH_PASSWORD, &userScheduler, MESH_PORT);
 mesh.onReceive(&receivedCallback);
 mesh.onNewConnection(&newConnectionCallback);
 mesh.onChangedConnections(&changedConnectionCallback);
 mesh.onNodeTimeAdjusted(&nodeTimeAdjustedCallback);
 mesh.onNodeDelayReceived(&delayReceivedCallback);
 userScheduler.addTask( taskSendMessage );
 taskSendMessage.enable();
 blinkNoNodes.set(BLINK_PERIOD, (mesh.getNodeList().size() + 1) * 2, []() {
 // If on, switch off, else switch on

 if (onFlag)
 onFlag = false;
 else
 onFlag = true;
 blinkNoNodes.delay(BLINK_DURATION);
 if (blinkNoNodes.isLastIteration()) {
 // Finished blinking. Reset task for next run
 // blink number of nodes (including this node) times
 blinkNoNodes.setIterations((mesh.getNodeList().size() + 1) * 2);
```

```
 // Calculate delay based on current mesh time and
 BLINK_PERIOD;
 // This results in blinks between nodes being synced
 blinkNoNodes.enableDelayed (BLINK_PERIOD - (mesh.getNodeTime() %
(BLINK_PERIOD*1000))/1000);
 }
 });
 userScheduler.addTask(blinkNoNodes);
 blinkNoNodes.enable();
 randomSeed(analogRead(A0));
}
void loop() {
 mesh.update();
 digitalWrite(LED, !onFlag);
}
void sendMessage() {
  // Read the humidity from the sensor.
    float humidity = dht11.readHumidity();

    // Read the temperature from the sensor.
    float temperature = dht11.readTemperature();

    // If the temperature and humidity readings were successful, print them to
the serial monitor.
    if (temperature != -1 && humidity != -1)
    {
        Serial.print("Temperature: ");
        Serial.print(temperature);
        Serial.println(" C");

        Serial.print("Humidity: ");
        Serial.print(humidity);
        Serial.println(" %");
    }
    else
    {
        // If the temperature or humidity reading failed, print an error
message.
        Serial.println("Error reading data");
    }

 String msg = "Hello from node 1";
 msg += mesh.getNodeId();
```

```
 msg += " myFreeMemory: " + String(ESP.getFreeHeap());
 mesh.sendBroadcast(msg);
 if (calc_delay) {
 SimpleList<uint32_t>::iterator node = nodes.begin();
 while (node != nodes.end()) {
 mesh.startDelayMeas(*node);

 node++;
 }
 calc_delay = false;
 }
 Serial.printf("Sending message: %s\n", msg.c_str());
 taskSendMessage.setInterval( random(TASK_SECOND * 1, TASK_SECOND * 5)); //
between 1 and 5 seconds
}
void receivedCallback(uint32_t from, String & msg) {
 Serial.printf("startHere: Received from %u msg=%s\n", from, msg.c_str());
}
void newConnectionCallback(uint32_t nodeId) {
 // Reset blink task
 onFlag = false;
 blinkNoNodes.setIterations((mesh.getNodeList().size() + 1) * 2);
 blinkNoNodes.enableDelayed(BLINK_PERIOD -
(mesh.getNodeTime() % (BLINK_PERIOD*1000))/1000);
 Serial.printf("--> startHere: New Connection, nodeId = %u\n", nodeId);
 Serial.printf("--> startHere: New Connection, %s\n",
mesh.subConnectionJson(true).c_str());
}
void changedConnectionCallback() {
 Serial.printf("Changed connections\n");
 // Reset blink task
 onFlag = false;
 blinkNoNodes.setIterations((mesh.getNodeList().size() + 1) * 2);
 blinkNoNodes.enableDelayed(BLINK_PERIOD - (mesh.getNodeTime() %
(BLINK_PERIOD*1000))/1000);

 nodes = mesh.getNodeList();
 Serial.printf("Num nodes: %d\n", nodes.size());
 Serial.printf("Connection list:");
 SimpleList<uint32_t>::iterator node = nodes.begin();
 while (node != nodes.end()) {
 Serial.printf(" %u", *node);
 node++;
```

```
 }
 Serial.println();
 calc_delay = true;
}
void nodeTimeAdjustedCallback(int32_t offset) {
 Serial.printf("Adjusted time %u. Offset = %d\n", mesh.getNodeTime(), offset);
}
void delayReceivedCallback(uint32_t from, int32_t delay) {
 Serial.printf("Delay to node %u is %d us\n", from, delay);
}
```

2. Program Node 2: MQ-2

```
 #include <painlessMesh.h>
#include <AsyncTCP.h>
#define LED 2
#define BLINK_PERIOD 3000
#define BLINK_DURATION 100
#define MESH_SSID "ESP32 Remote Lab"
#define MESH_PASSWORD "1234567890"
#define MESH_PORT 5555
#define mq2Pin 32   // Pin analog untuk sensor MQ-2
#include <Wire.h>
void sendMessage();

void receivedCallback(uint32_t from, String & msg);
void newConnectionCallback(uint32_t nodeId);
void changedConnectionCallback();
void nodeTimeAdjustedCallback(int32_t offset);
void delayReceivedCallback(uint32_t from, int32_t delay);
Scheduler userScheduler; // to control your personal task
painlessMesh mesh;
bool calc_delay = false;
SimpleList<uint32_t> nodes;
void sendMessage() ; // Prototype
Task taskSendMessage( TASK_SECOND * 1, TASK_FOREVER,
&sendMessage ); // start with a one second interval
// Task to blink the number of nodes
Task blinkNoNodes;
bool onFlag = false;
void setup() {
 Serial.begin(9600);
 pinMode(mq2Pin , INPUT);
```

```
 pinMode(LED, OUTPUT);
 mesh.setDebugMsgTypes(ERROR | DEBUG); // set before init() so that you can
see error messages
 mesh.init(MESH_SSID, MESH_PASSWORD, &userScheduler, MESH_PORT);
 mesh.onReceive(&receivedCallback);
 mesh.onNewConnection(&newConnectionCallback);
 mesh.onChangedConnections(&changedConnectionCallback);
 mesh.onNodeTimeAdjusted(&nodeTimeAdjustedCallback);
 mesh.onNodeDelayReceived(&delayReceivedCallback);
 userScheduler.addTask( taskSendMessage );
 taskSendMessage.enable();
 blinkNoNodes.set(BLINK_PERIOD, (mesh.getNodeList().size() + 1) * 2, []() {
 // If on, switch off, else switch on

 if (onFlag)
 onFlag = false;
 else
 onFlag = true;
 blinkNoNodes.delay(BLINK_DURATION);
 if (blinkNoNodes.isLastIteration()) {
 // Finished blinking. Reset task for next run
 // blink number of nodes (including this node) times
 blinkNoNodes.setIterations((mesh.getNodeList().size() + 1) * 2);
 // Calculate delay based on current mesh time and
 BLINK_PERIOD;
 // This results in blinks between nodes being synced
 blinkNoNodes.enableDelayed (BLINK_PERIOD - (mesh.getNodeTime() %
(BLINK_PERIOD*1000))/1000);
 }
 });
 userScheduler.addTask(blinkNoNodes);
 blinkNoNodes.enable();
 randomSeed(analogRead(A0));
}
void loop() {
 mesh.update();
 digitalWrite(LED, !onFlag);
}
void sendMessage() {
  int sensorValue = analogRead(mq2Pin); // Membaca nilai sensor analog

  float voltage = sensorValue * (5.0 / 1023.0); // Mengonversi nilai sensor
menjadi tegangan (5V adalah tegangan referensi Arduino)
```

```
  // Menghitung konsentrasi gas menggunakan rumus yang sesuai dengan sensor MQ-2
  float gasResistance = ((5.0 - voltage) / voltage) * 10.0; // Menggunakan faktor 10.0 untuk mengkoreksi nilai

  // Menampilkan hasil ke Serial Monitor
  Serial.print("Sensor Value: ");
  Serial.println(sensorValue);
  delay(2000);
  Serial.print("Gas Resistance: ");
  Serial.print(gasResistance);
  Serial.println(" KΩ");
 String msg = "Hello from node 3";
 msg += mesh.getNodeId();
 msg += " myFreeMemory: " + String(ESP.getFreeHeap());
 mesh.sendBroadcast(msg);
 if (calc_delay) {
 SimpleList<uint32_t>::iterator node = nodes.begin();
 while (node != nodes.end()) {
 mesh.startDelayMeas(*node);

 node++;
 }
 calc_delay = false;
 }
 Serial.printf("Sending message: %s\n", msg.c_str());
 taskSendMessage.setInterval( random(TASK_SECOND * 1, TASK_SECOND * 5)); // between 1 and 5 seconds
}
void receivedCallback(uint32_t from, String & msg) {
 Serial.printf("startHere: Received from %u msg=%s\n", from, msg.c_str());
}
void newConnectionCallback(uint32_t nodeId) {
 // Reset blink task
 onFlag = false;
 blinkNoNodes.setIterations((mesh.getNodeList().size() + 1) * 2);
 blinkNoNodes.enableDelayed(BLINK_PERIOD - (mesh.getNodeTime() % (BLINK_PERIOD*1000))/1000);
 Serial.printf("--> startHere: New Connection, nodeId = %u\n", nodeId);
 Serial.printf("--> startHere: New Connection, %s\n", mesh.subConnectionJson(true).c_str());
}
```

```
void changedConnectionCallback() {
 Serial.printf("Changed connections\n");
 // Reset blink task
 onFlag = false;
 blinkNoNodes.setIterations((mesh.getNodeList().size() + 1) * 2);
 blinkNoNodes.enableDelayed(BLINK_PERIOD - (mesh.getNodeTime() %
(BLINK_PERIOD*1000))/1000);

 nodes = mesh.getNodeList();
 Serial.printf("Num nodes: %d\n", nodes.size());
 Serial.printf("Connection list:");
 SimpleList<uint32_t>::iterator node = nodes.begin();
 while (node != nodes.end()) {
 Serial.printf(" %u", *node);
 node++;
 }
 Serial.println();
 calc_delay = true;
}
void nodeTimeAdjustedCallback(int32_t offset) {
 Serial.printf("Adjusted time %u. Offset = %d\n", mesh.getNodeTime(), offset);
}
void delayReceivedCallback(uint32_t from, int32_t delay) {
 Serial.printf("Delay to node %u is %d us\n", from, delay);
}
```

3. Program Node 3: Soil Sensor

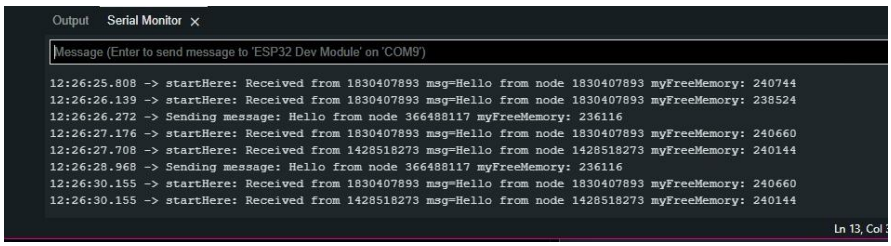
```
#include <painlessMesh.h>
#include <AsyncTCP.h>
#define LED 2
#define BLINK_PERIOD 3000
#define BLINK_DURATION 100
#define MESH_SSID "ESP32 Remote Lab"
#define MESH_PASSWORD "1234567890"
#define MESH_PORT 5555
int SensorPin = 32;// deklarasi pin analog yg dipakai
int soilMoistureValue; // menyimpan nilai analog dari sensor ke esp32
int soilmoisturepercent; // nilai yg diperoleh dalam bentuk persen setelah
dimaping
// Prototypes
void sendMessage();
```

```
void receivedCallback(uint32_t from, String & msg);
void newConnectionCallback(uint32_t nodeId);
void changedConnectionCallback();
void nodeTimeAdjustedCallback(int32_t offset);
void delayReceivedCallback(uint32_t from, int32_t delay);
Scheduler userScheduler; // to control your personal task
painlessMesh mesh;
bool calc_delay = false;
SimpleList<uint32_t> nodes;
void sendMessage() ; // Prototype
Task taskSendMessage( TASK_SECOND * 1, TASK_FOREVER,
&sendMessage ); // start with a one second interval
// Task to blink the number of nodes
Task blinkNoNodes;
bool onFlag = false;
void setup() {
 Serial.begin(9600);
 pinMode(LED, OUTPUT);
 mesh.setDebugMsgTypes(ERROR | DEBUG); // set before init() so that you can
see error messages
 mesh.init(MESH_SSID, MESH_PASSWORD, &userScheduler, MESH_PORT);
 mesh.onReceive(&receivedCallback);
 mesh.onNewConnection(&newConnectionCallback);
 mesh.onChangedConnections(&changedConnectionCallback);
 mesh.onNodeTimeAdjusted(&nodeTimeAdjustedCallback);
 mesh.onNodeDelayReceived(&delayReceivedCallback);
 userScheduler.addTask( taskSendMessage );
 taskSendMessage.enable();
 blinkNoNodes.set(BLINK_PERIOD, (mesh.getNodeList().size() + 1) * 2, []() {
 // If on, switch off, else switch on

 if (onFlag)
 onFlag = false;
 else
 onFlag = true;
 blinkNoNodes.delay(BLINK_DURATION);
 if (blinkNoNodes.isLastIteration()) {
 // Finished blinking. Reset task for next run
 // blink number of nodes (including this node) times
 blinkNoNodes.setIterations((mesh.getNodeList().size() + 1) * 2);
 // Calculate delay based on current mesh time and
 BLINK_PERIOD;
 // This results in blinks between nodes being synced
```

```
 blinkNoNodes.enableDelayed (BLINK_PERIOD - (mesh.getNodeTime() %
(BLINK_PERIOD*1000))/1000);
 }
 });
 userScheduler.addTask(blinkNoNodes);
 blinkNoNodes.enable();
 randomSeed(analogRead(A0));
}
void loop() {
 mesh.update();
 digitalWrite(LED, !onFlag);
}
void sendMessage() {
  // Read the humidity from the sensor.
      soilMoistureValue = analogRead(SensorPin);
        Serial.print("Nilai analog = ");
          Serial.print(soilMoistureValue);
           soilmoisturepercent = map(soilMoistureValue, 4095, 0, 0, 100);

        Serial.print(" Presentase kelembaban tanah= ");
       Serial.print(soilmoisturepercent);
      Serial.println("% ");
 String msg = "Hello from node 2";
 msg += mesh.getNodeId();
 msg += " myFreeMemory: " + String(ESP.getFreeHeap());
 mesh.sendBroadcast(msg);
 if (calc_delay) {
 SimpleList<uint32_t>::iterator node = nodes.begin();
 while (node != nodes.end()) {
 mesh.startDelayMeas(*node);

 node++;
 }
 calc_delay = false;
 }
 Serial.printf("Sending message: %s\n", msg.c_str());
 taskSendMessage.setInterval( random(TASK_SECOND * 1, TASK_SECOND * 5)); //
between 1 and 5 seconds
}
void receivedCallback(uint32_t from, String & msg) {
 Serial.printf("startHere: Received from %u msg=%s\n", from, msg.c_str());
}
void newConnectionCallback(uint32_t nodeId) {
```

```
 // Reset blink task
 onFlag = false;
 blinkNoNodes.setIterations((mesh.getNodeList().size() + 1) * 2);
 blinkNoNodes.enableDelayed(BLINK_PERIOD -
(mesh.getNodeTime() % (BLINK_PERIOD*1000))/1000);
 Serial.printf("--> startHere: New Connection, nodeId = %u\n", nodeId);
 Serial.printf("--> startHere: New Connection, %s\n",
mesh.subConnectionJson(true).c_str());
}
void changedConnectionCallback() {
 Serial.printf("Changed connections\n");
 // Reset blink task
 onFlag = false;
 blinkNoNodes.setIterations((mesh.getNodeList().size() + 1) * 2);
 blinkNoNodes.enableDelayed(BLINK_PERIOD - (mesh.getNodeTime() %
(BLINK_PERIOD*1000))/1000);

 nodes = mesh.getNodeList();
 Serial.printf("Num nodes: %d\n", nodes.size());
 Serial.printf("Connection list:");
 SimpleList<uint32_t>::iterator node = nodes.begin();
 while (node != nodes.end()) {
 Serial.printf(" %u", *node);
 node++;
 }
 Serial.println();
 calc_delay = true;
}
void nodeTimeAdjustedCallback(int32_t offset) {
 Serial.printf("Adjusted time %u. Offset = %d\n", mesh.getNodeTime(), offset);
}
void delayReceivedCallback(uint32_t from, int32_t delay) {
 Serial.printf("Delay to node %u is %d us\n", from, delay);
}
```

Hasil Program



V. Kesimpulan

Kesimpulannya hampir sama seperti pada Job 3, tetapi pada masing-masing node terdapat data sensor yang dikirim.

JOB 5 Pemrograman ESP32 Menggunakan OTA

I. Tujuan

Peserta didik dapat memprogram ESP32 dengan bertukar data sensor yang dimiliki antar node menggunakan topologi Wi-Fi Mesh Network dengan cara mengupload program menggunakan OTA (Over The Air)

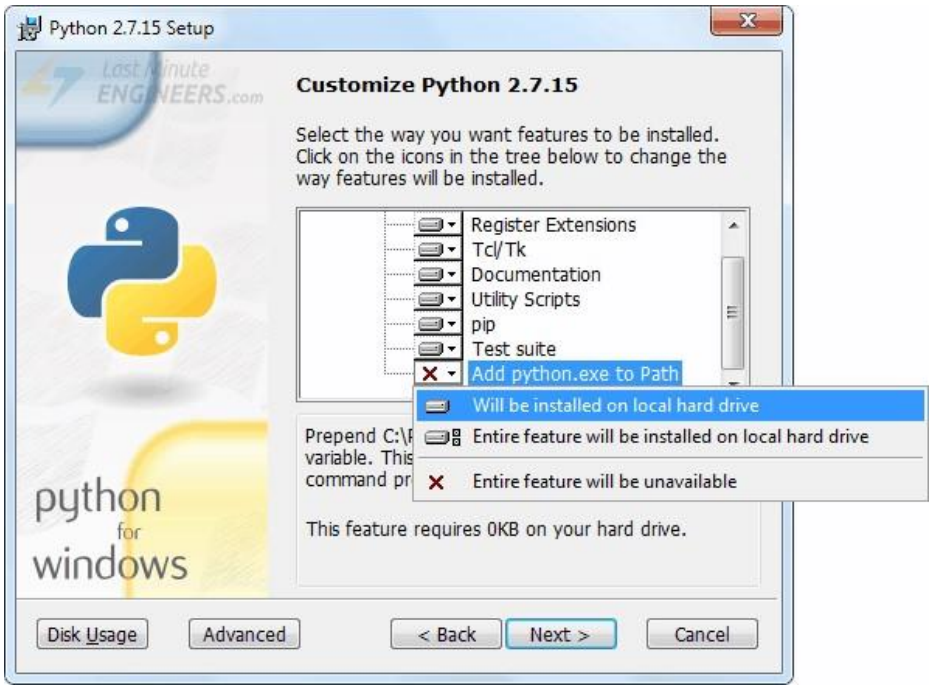
II. Alat dan Bahan

Adapun yang harus disediakan yaitu:

1. Komputer/Laptop dengan koneksi internet.
2. *Software* Arduino IDE.
3. Aplikasi TeamViewer

III. Langkah Kerja

1. Langkah yang digunakan berbeda dengan praktikum sebelumnya, yang pertama unduh dan install *software* Python 2.7.x pada komputer/laptop
2. Saat menginstall *software* pastikan opsi pada bagian **Add Python.exe to Path** diaktifkan



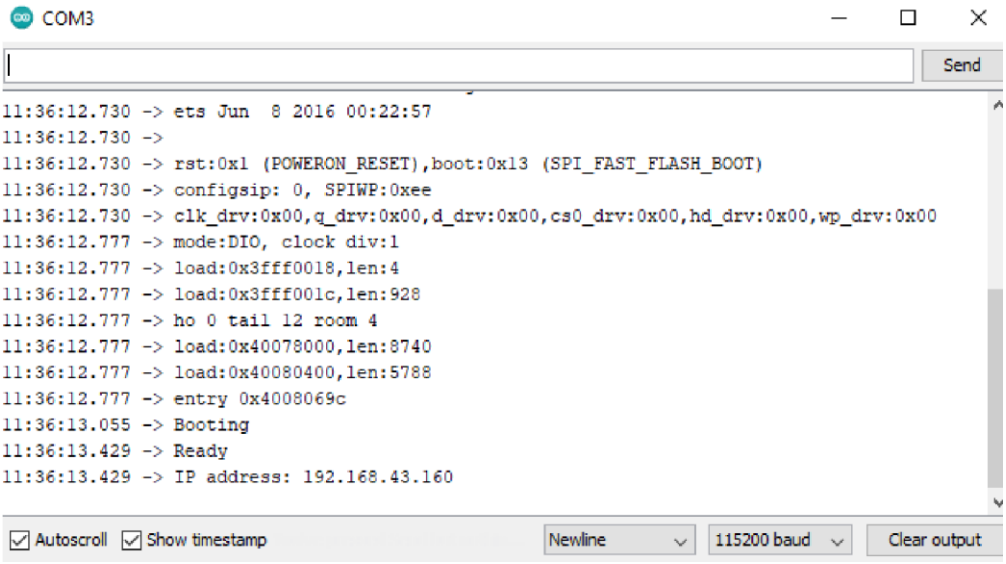
3. Pada Arduino IDE tidak memiliki untuk memperbaharui OTA sehingga harus membuat firmware OTA pada ESP32. Untuk memperbaharui

firmware harus melakukan serial interface terlebih dahulu, langkahnya yaitu buka Arduino IDE klik **File > Examples > ArduinoOTA > BasicOTA**.

4. Sebelum mengupload program, sesuaikan bagian dibawah ini dengan nama koneksi jaringan yang tersedia

```
const char* ssid = ".........."; const
char* password = "..........";
```

5. Setelah berhasil diupload buka serial monitor, sampai hasil seperti gambar di bawah



6. Selanjutnya upload program ESP Wi-Fi Mesh Network menggunakan OTA Program:

```
#include <WiFi.h>
#include <ESPmDNS.h>
#include <WiFiUdp.h>
#include <ArduinoOTA.h>
#include <painlessMesh.h>

#define   LED             2       // GPIO number of
connected LED, ON ESP-12 IS GPIO2
#define   BLINK_PERIOD    3000 // milliseconds until cycle
repeat
#define   BLINK_DURATION  100  // milliseconds LED is on for
#define   MESH_SSID       "ESP32 Remote Lab"
```

```
#define   MESH_PASSWORD   "1234567890"
#define   MESH_PORT       5555
 const char* ssid = ". . . . . .";
const char* password = ". . . . . .";
 void sendMessage();  void
receivedCallback(uint32_t from, String & msg); void
newConnectionCallback(uint32_t nodeId);
void changedConnectionCallback();  void
nodeTimeAdjustedCallback(int32_t offset);  void
delayReceivedCallback(uint32_t from, int32_t delay);

Scheduler     userScheduler; // to control your personal
task painlessMesh  mesh;

bool calc_delay = false; SimpleList<uint32_t>
nodes;

void sendMessage() ; // Prototype
Task taskSendMessage( TASK_SECOND * 1, TASK_FOREVER,
&sendMessage ); // start with a one second interval

// Task to blink the number of nodes
Task blinkNoNodes; bool onFlag =
false;
  void setup() {
Serial.begin(115200);
  Serial.println("Booting");
  WiFi.mode(WIFI_STA);
WiFi.begin(ssid, password);
  while (WiFi.waitForConnectResult() != WL_CONNECTED) {
Serial.println("Connection Failed! Rebooting...");
delay(5000);     ESP.restart();

  }
```

```
  // Port defaults to 3232
  // ArduinoOTA.setPort(3232);

  // Hostname defaults to esp3232-[MAC]
  // ArduinoOTA.setHostname("myesp32");

  // No authentication by default
  // ArduinoOTA.setPassword("admin");

  // Password can be set with it's md5 value as well
  // MD5(admin) = 21232f297a57a5a743894a0e4a801fc3
  //
ArduinoOTA.setPasswordHash("21232f297a57a5a743894a0e4a801fc3
");

  ArduinoOTA
    .onStart([]() {       String type;
if (ArduinoOTA.getCommand() == U_FLASH)
type = "sketch";       else // U_SPIFFS
type = "filesystem";

      // NOTE: if updating SPIFFS this would be the place to
unmount SPIFFS using SPIFFS.end()
Serial.println("Start updating " + type);
    })
    .onEnd([]() {
      Serial.println("\nEnd");
    })
    .onProgress([](unsigned int progress, unsigned int
total) {
      Serial.printf("Progress: %u%%\r", (progress / (total /
100)));
    })
    .onError([](ota_error_t error) {
Serial.printf("Error[%u]: ", error);       if (error ==
OTA_AUTH_ERROR) Serial.println("Auth
```

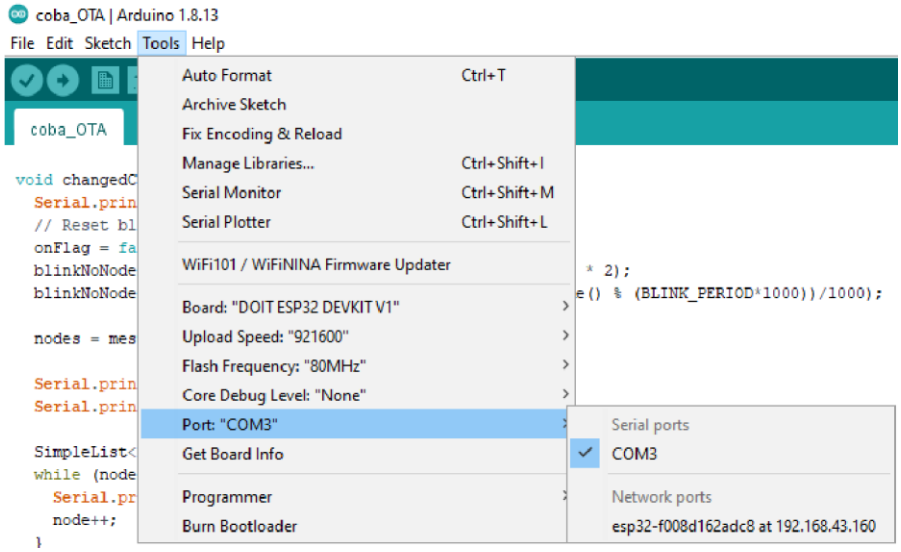
```
Failed");
      else if (error == OTA_BEGIN_ERROR)
Serial.println("Begin Failed");       else
if (error == OTA_CONNECT_ERROR)
Serial.println("Connect Failed");
else if (error == OTA_RECEIVE_ERROR)
Serial.println("Receive Failed");
      else if (error == OTA_END_ERROR) Serial.println("End
Failed");
    });

  ArduinoOTA.begin();

  Serial.println("Ready");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
   mesh.setDebugMsgTypes(ERROR | DEBUG);  // set
before init() so that you can see error messages

  mesh.init(MESH_SSID, MESH_PASSWORD, &userScheduler,
MESH_PORT);   mesh.onReceive(&receivedCallback);
mesh.onNewConnection(&newConnectionCallback);
mesh.onChangedConnections(&changedConnectionCallback);
mesh.onNodeTimeAdjusted(&nodeTimeAdjustedCallback);
mesh.onNodeDelayReceived(&delayReceivedCallback);
   userScheduler.addTask( taskSendMessage
);   taskSendMessage.enable();
   blinkNoNodes.set(BLINK_PERIOD,
(mesh.getNodeList().size()
+ 1) * 2, []() {
      // If on, switch off, else switch on
if (onFlag)         onFlag = false;
else
        onFlag = true;
blinkNoNodes.delay(BLINK_DURATION);
```

```
       if (blinkNoNodes.isLastIteration()) {
// Finished blinking. Reset task for next run
        // blink number of nodes (including this node) times

blinkNoNodes.setIterations((mesh.getNodeList().size() + 1) *
2);
        // Calculate delay based on current mesh time and
BLINK_PERIOD
        // This results in blinks between nodes being synced
blinkNoNodes.enableDelayed(BLINK_PERIOD -
(mesh.getNodeTime() %
(BLINK_PERIOD*1000))/1000);
      }   });
userScheduler.addTask(blinkNoNodes);
blinkNoNodes.enable();

randomSeed(analogRead(A0));

}  void loop() {
ArduinoOTA.handle();
mesh.update();
digitalWrite(LED, !onFlag);
}
void sendMessage() {   String msg = "Hello from node ";
msg += mesh.getNodeId();   msg += " myFreeMemory: " +
String(ESP.getFreeHeap());   mesh.sendBroadcast(msg);

  if (calc_delay) {
    SimpleList<uint32_t>::iterator node = nodes.begin();
while (node != nodes.end()) {
mesh.startDelayMeas(*node);       node++;     }
calc_delay = false;
  }

  Serial.printf("Sending message: %s\n", msg.c_str());
```

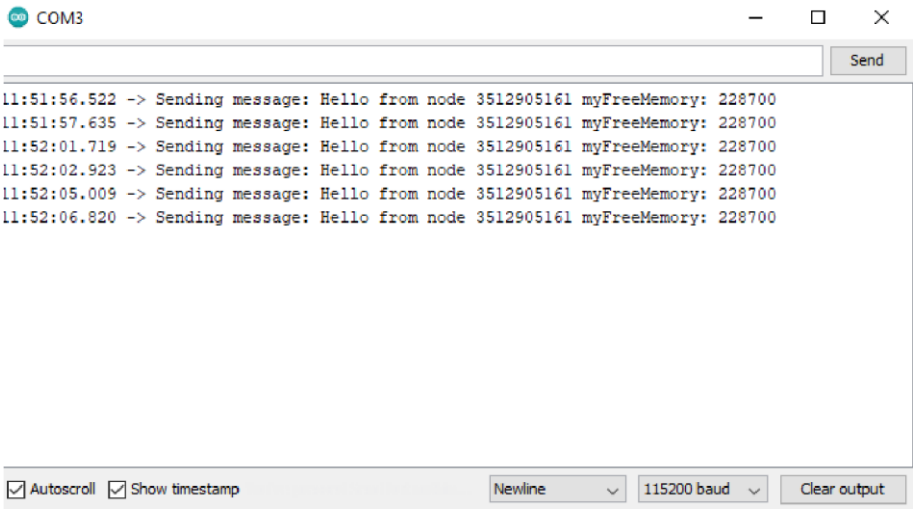
```
  taskSendMessage.setInterval( random(TASK_SECOND * 1,
TASK_SECOND * 5));  // between 1 and 5 seconds
}

void receivedCallback(uint32_t from, String & msg) {
Serial.printf("startHere: Received from %u msg=%s\n", from,
msg.c_str());
}  void newConnectionCallback(uint32_t nodeId) {   // Reset
blink task   onFlag = false;
blinkNoNodes.setIterations((mesh.getNodeList().size() + 1)
*   2);   blinkNoNodes.enableDelayed(BLINK_PERIOD
- (mesh.getNodeTime() %
(BLINK_PERIOD*1000))/1000);

  Serial.printf("--> startHere: New Connection, nodeId =
%u\n", nodeId);
  Serial.printf("--> startHere: New Connection, %s\n",
mesh.subConnectionJson(true).c_str());
}  void
changedConnectionCallback() {
Serial.printf("Changed
connections\n");
  // Reset blink task   onFlag = false;
blinkNoNodes.setIterations((mesh.getNodeList().size() + 1)
*   2);   blinkNoNodes.enableDelayed(BLINK_PERIOD
- (mesh.getNodeTime() %
(BLINK_PERIOD*1000))/1000);
    nodes =
mesh.getNodeList();

  Serial.printf("Num nodes: %d\n", nodes.size());
Serial.printf("Connection list:");
```

```
  SimpleList<uint32_t>::iterator node = nodes.begin();
while (node != nodes.end()) {     Serial.printf(" %u",
*node);     node++;
  }
  Serial.println();
calc_delay = true;
}
void nodeTimeAdjustedCallback(int32_t offset) {
Serial.printf("Adjusted time %u. Offset = %d\n",
mesh.getNodeTime(), offset);
}
void delayReceivedCallback(uint32_t from, int32_t delay) {
Serial.printf("Delay to node %u is %d us\n", from, delay);
}
```

7. Untuk mengupload menggunakan OTA tidak menggunakan port yang tersedia, melainkan menggunakan **esp32-xxxxxx at your_esp_ip_address**. Jika tidak ada maka restart Arduino IDE



8. Jika menggunakan tidak dapat melihat hasil melalui serial monitor, maka langkah yang digunakan untuk melihat hasil dengan mengubah port dengan klik **Tools > Ports > (Pilih Port yang digunakan)** setelah itu dapat melihat hasil di serial monitor.



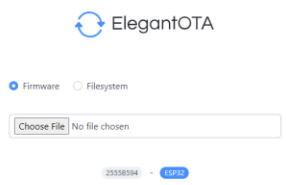
IV. Hasil Praktikum

1. Program

```
#include <Arduino.h>
#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <AsyncElegantOTA.h>

const char* ssid = "Lantai 2";
const char* password = "25mei2023";

AsyncWebServer server(80);

void setup(void) {

  Serial.begin(9600);
  WiFi.begin(ssid, password);
  Serial.println("");

  //wait fot connection
  while (WiFi.status()!=WL_CONNECTED){
  delay(500);
  Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to");
  Serial.println(ssid);
```

```
  Serial.print("IP addres ");
  Serial.println(WiFi.localIP());

  server.on("/" , HTTP_GET, [] (AsyncWebServerRequest *request) {
    request->send(200, "text/plain", "ESP32 OTA (Over The Air).");
  });
  AsyncElegantOTA.begin(&server);
  server.begin();
  Serial.println(" HTTP Server Started");
}
void loop(void){
  AsyncElegantOTA.loop();
  digitalWrite(2, HIGH);
  delay(1000);
  digitalWrite(2, LOW);
  delay(1000);
}
```

2. Hasil program



VI. Kesimpulan

Setelah melakukan percobaan Job 5 ini, kami bisa melakukan uploading program ke ESP32 tanpa harus port ESP32 terpasang di perangkat.

IOT DENGAN DASHBOARD FIREBASE

1. Program

```
#include <FirebaseESP32.h>
#include <WiFi.h>

#define FIREBASE_HOST "https://node-mq-2-default-rtdb.firebaseio.com/"
#define FIREBASE_AUTH "4uBddN3Kxe8umvdg4qfPsF15WcAcuYK9YJD7BeZV"

#define WIFI_SSID "Zahran"
#define WIFI_PASSWORD "Zahran007"

FirebaseData fbdo; //fbdo adalah variabel.

#define PinDigital 4 // mendefinisikan pin yang digunakan adalah pin Digital
int NilaiDigital;

void setup() {
  Serial.begin(9600);
  pinMode(PinDigital, INPUT); //mode pada pin D4 dijadikan sebagai input

  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Menghubungkan Wi-Fi");
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(300);
  }
  Serial.println();
  Serial.print("terhubung dengan WiFi IP: ");
  Serial.println(WiFi.localIP());
  Serial.println();

  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
}

void loop() {
  NilaiDigital = digitalRead(PinDigital); // membaca nilai digital

  Serial.print("Nilai Output Digital = ");
  Serial.println(NilaiDigital);
  //Aktif LOW = Jika Ada Asap Maka Nilai nya 0 jika tidak ada Asap maka nilai
Nya 1
```

```
  //Proses Kirim Data
  Firebase.setFloat(fbdo, "/Nilai_Asap", NilaiDigital);

  if(NilaiDigital==0){
    Firebase.setString(fbdo, "/Kondisi", "Ada Asap");
  } else {
    Firebase.setString(fbdo, "/Kondisi", "Aman");
  }

  delay(1000);
}
```

2. Hasil Program