

1. Sentrale forskjeller mellom Arduino Uno og ESP32 (20%)

- Hva er de sentrale forskjellene mellom en ESP32 og en Arduino Uno? Fordeler/ulemper? Spørsmålene under har ikke noe med disse forskjellene å gjøre (dere kan altså ikke gi svar her på det som er spurt om i de neste spørsmålene).

En ESP32 kan alt en Arduino Uno kan, men kan gjøre mye mer og med bedre tekniske spesifikasjoner. Fordelen med ESP32 er at man kan programmere mer og bruke det til flere applikasjoner, WiFi og Bluetooth funksjonalitet. Det går derimot kanskje på bekostning av brukervennlighet i og med at man må installere et ekstra bibliotek. I grunnen dog er det ingen ulempe.

- Hva er maks CPU-frekvens for ESP32-ene og hva er i følge Arduino IDE lavest anbefalte frekvens for Wi-Fi bruk? Hvor mange kjerner har CPU-en til ESP32-en vi bruker? Hvorfor er det bedre med høyere CPU-frekvens og flere kjerner? Hva er de negative sidene ved det?

For ESP32 er 240MHz maks klokkehastighet/CPU-frekvens med 2.4 GHz for Bluetooth og Wi-Fi. Arduino IDE støtter kun 2.4 GHz for Wi-Fi bruk. Det er to kjerner i ESP32-en. Høyere CPU-frekvens betyr en raskere og sterkere CPU noe som kan utføre operasjoner bedre. Flere kjerner lar en å kjøre flere programmere/void loops samtidig. Det kan være kanskje vanskelig for en nybegynner å bruke.

- Hva slags praktisk konsekvens utgjør <<Upload Speed>>, som standard er satt til <<921600>> i Arduino IDE (test det gjerne ut)? Hva kan etterhvert være konsekvensene av at en setter <<Upload Speed>> til maksverdien?

Opplastningshastighet lar oss bestemme hvor rask Arduino IDEen skal «uplade» arduino-skriptet til brettet/ESP32en. Med maks Upload Speed kan man laste opp programmene veldig rask.

- Har <<Port>>/COMPORT et navn når en ESP32 er koblet til PC-en? Utenom <<COM>> etterfulgt av et tall som dere kanskje har sett før? Har en Arduino Uno det? Hvorfor kan dette være viktig å være oppmerksom på?

Det er flere porter fra /dev/ folderen, altså folderen som tar inn eksterne dingser som tilkobles datamaskinen. Jeg ser wlan, bluetooth, og andre koblinger som jeg hadde før som trådløse hodetelefoner. Den tar inn usb serial for min del.

- Har ESP32-en støtte for UART, I2C og SPI? Hvorfor er dette viktig mtp. andre mikrokontrollere og teknologi forøvrig?

Ja, ESP32 har den støtten. UART står for universal asynchronous receiver transmitter, en funksjonalitet som lar ESP32 kommunisere med andre dingser. SPI lar kortavstandskommunikasjon med IC (integreerte kretser), og står for serial peripheral

interface. I2C står for inter integrated communication og lar kommunikasjon innad i brettet mellom ICs.

2. Grunnleggende signalteori (ADC, DAC, PWM) (10%)

- Både en Arduino Uno og en ESP32 har en ADC. Kvalitetene på disse to ADC-ene er derimot forskjellig. Hva er den ene mest sentrale forskjellen? Hva kan derfor være konsekvensen av å bruke en Arduino Uno, istedenfor en ESP32, til å måle et signal?

Arduino Uno har en 10-bit ADC (avhengig av versjon e.g. R4 har 14-bit) mens en ESP32 har flere 12-bit ADC. Høyere bit lar bedre «resolution» på analog signalet siden det er flere diskrete verdier som kan utforme det analoge signalet på. Arduino Uno fører til dårligere kvalitet på målingen av analoge signaler.

- Har en Arduino Uno en DAC? Har ESP32 det? Forklart kort hva en DAC er og hvorfor våre behov heller kan dekkes med PWM-teknologi.

En DAC er en digital-til-analog omformer. ESP32 har to 8-bit, mens en Arduino Uno har ingen DAC (fleste versjoner ingen men R4 har). PWM-teknologi lar 16-bit for DAC i ESP32. Dette er bedre enn 8-bit for Arduino Uno.

- I dine egne ord: Hvordan fungerer/genererer PWM-teknologi et analogt signal? Hent en figur fra internettet og bruk den til å forklare.

En PWM-signal sendes til en LPF-krets (low pass filter e.g. inngang-R-utgang-C-GND) som lar lav frekvens signaler gjennom. Dersom bredden er langt, vil det korresponderende analoge signalet ha en høyverdi

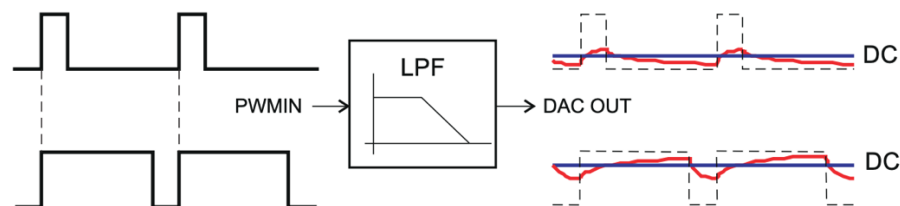


Figure 1-2. PWM DAC Implementation

- Hvorfor må du i noen tilfeller være oppmerksom på hvordan du bruker flere PWM-utganger samtidig på en ESP32?

Det er noen biblioteker som servo-biblioteket som tar oppsettet av pwm signalet i seg selv. Dette kan kollidere med egen kode/oppsett. For eksempel kan man bruke kanal 1 men den kan overskrives av senere kode av servo-biblioteket.

- Hvordan bytter du PWM-oppløsning på en ESP32 (oppgi den/de spesifikke kommandolinjen)? Hva kan den byttes i mellom og kan alle utgangene ha samme oppløsning, samtidig? [Denne lenken kan være nyttig.](#)

```
void analogWriteResolution(uint8_t pin, uint8_t resolution);
```

- `pin` select LEDC pin.
- `resolution` select resolution for LEDC channel
 - range is 1-14 bits (1-20 bits for ESP32).

I vår ESP32 er det 16 kanaler, 1 kHz frekvens til pwm avhengig av kanal, og opptil muligens 16-bit oppløsning/resolusjon avhengig av kanal.

3. Tingenes internett (IoT) (10%)

- Diskuter kort hvorfor en ESP32 er bedre egnet til IoT-applikasjoner, enn en Arduino Uno (her er det flere punkter enn bare «det åpenbare»). Denne lenken kan gi noen svar.

ESP32 har innebygd Wi-Fi og bluetooth funksjonalitet samt kommunikasjon til andre mikrokontrollere som lar mikrokontrolleren å være egnet til IoT-applikasjoner. I tillegg er ESP32 3.3 volt med lav-effekt modus noe som er viktig mtp. hvordan småelektronikken forbruker strøm og elektrisk energi på. Det blir billig i det lange løpet der i IoT-applikasjoner flere ESP32 kobles. ESP32 er også robust i forskjellige omgivelser (operativområde) noe som forsikrer at småelektronikken fungerer over lang tid.

- Tenk deg frem til minst 5 forskjellige problemer/applikasjoner/bruksscenarier du mener kan løses med en ESP32. Et eksempel kan være automatisk overvåking og kontroll av miljøet innendørs i kontorbygg. Denne lenken kan gi litt inspirasjon.
 1. Styring av temperatur i matlaging
 2. Kontroll av lys i smartehus/grønnehus
 3. Kontroll av varmeovn i smarehus
 4. Overvåking av luft for luftvern
 5. Overvåking av helse via puls, pust, og temperatur
 6. Miljøovervåking, e.g., nordpolen
- Industriell IoT, eller såkalt Industri 4.0 også kjent som den fjerde industrielle revolusjon, er et svært forskningsområde på anvendelser av IoT i industrien. Søk opp disse begrepene og forklart kort hva slags betydning dette forskningsområde kan ha for fremtiden vår.

Den har en stor betydning for fremtiden i og med IIoT har muligheten til å effektivisere og øke produksjonen. Den forbedrer altså økonomien. I tillegg så kan man med IIoT minske feil som skjer i produksjonslinjer og levere produkter til kundene bedre. Den kan tenkes på som en forbedret SCADA.

- Moderne Wi-Fi rutere kommer vanligvis forhåndsjustert til to forskjellige frekvensområder, hvilket er disse? Hvilket støtter ESP32-en? Hva skjer om du forsøker å koble til et Wi-Fi med frekvens som ESP32-en ikke støtter (test dette ut gjennom programmeringsoppgavene og kommenter her)?

Moderne Wi-Fi rutere støtter 2.4 GHz og 5 GHz. ESP32 bruker 2.4 GHz.

```
5
6  Servo servo1;
7
8  const char *ssid = "ESP32_test";
9  const char *password = "MidjoSkyen2";
10
11  const int LED = 18;
12  const int BUZZ = 23;
13  const int channelLED = 0;
14  const int channelBUZZ = 1;
15  const int channelSERVO1 = 2;
16
17  void setup()
18  {
19
20
```

PROBLEMS 7 OUTPUT TERMINAL GITLENS

> ✓ TERMINAL

Writing at 0x000bd14c... (96 %)
Writing at 0x000c2df5... (100 %)
Wrote 734160 bytes (476157 compressed) at 0x00010000 in 12.3 seconds (effective 473.4 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
--- Terminal on /dev/cu.usbserial-10 | 9600 8-N-1
--- Available filters and text transformations: colorize, debug, default, direct, esp32_exception_decoder, hexlify, send_on_enter, time
--- More details at <https://bit.ly/pio-monitor-filters>
--- Quit: Ctrl+C | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H
Trying to connect to
ESP32_test
.....
WiFi connected successfully
Got IP: 192.168.50.44
□

Fig 1. ESP32 kobler seg til 2.4 Ghz

```
Servo servo1;

const char *ssid = "ESP32_test_5G";
const char *password = "MidjoSkyen5";

const int LED = 18;
const int BUZZ = 23;
const int channelLED = 0;
const int channelBUZZ = 1;
const int channelSERVO1 = 2;

void setup()
{
  /*
   * if (WiFi.status() == WL_CONNECTED) {
   */
}

PROBLEMS 7 OUTPUT TERMINAL GITLENS
```

TERMINAL

Writing at 0x000acc38... (86 %)
Writing at 0x000b1ff7... (90 %)
Writing at 0x000b7d18... (93 %)
Writing at 0x000bd143... (96 %)
Writing at 0x000c2deb... (100 %)
Wrote 734160 bytes (476164 compressed) at 0x00010000 in 12.4 seconds (effective 473.4 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
--- Terminal on /dev/cu.usbserial-10 | 9600 8-N-1
--- Available filters and text transformations: colorize, debug, default, direct, esp32_exception_decoder, hexlify, send_on_enter, time
--- More details at <https://bit.ly/pio-monitor-filters>
--- Quit: Ctrl+C | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H
Trying to connect to
ESP32_test_5G
.....
□

Fig 2. ESP32 klarer ikke å koble til 5GHz (ventet for 1 minutt)

Programmering og oppkobling (60%)

4. Installasjon og test av ESP32 i Arduino IDE (20%)

- Installer ESP32 i Arduino IDE

Installert både Arduino IDE og platformIO ved vscode.

- Installer ESP32-driver fra Silabs

Mac M1 erkjenner ESP32. Den har ESP32-driveren allerede installert. Den

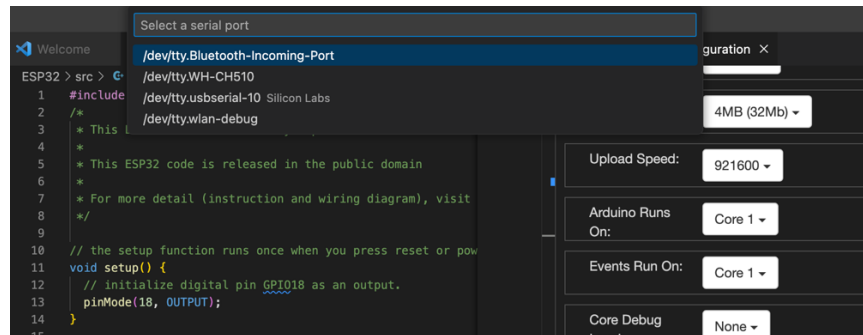


Fig 3. ESP32-driver fra silicon labs

- Blink et LED-lys

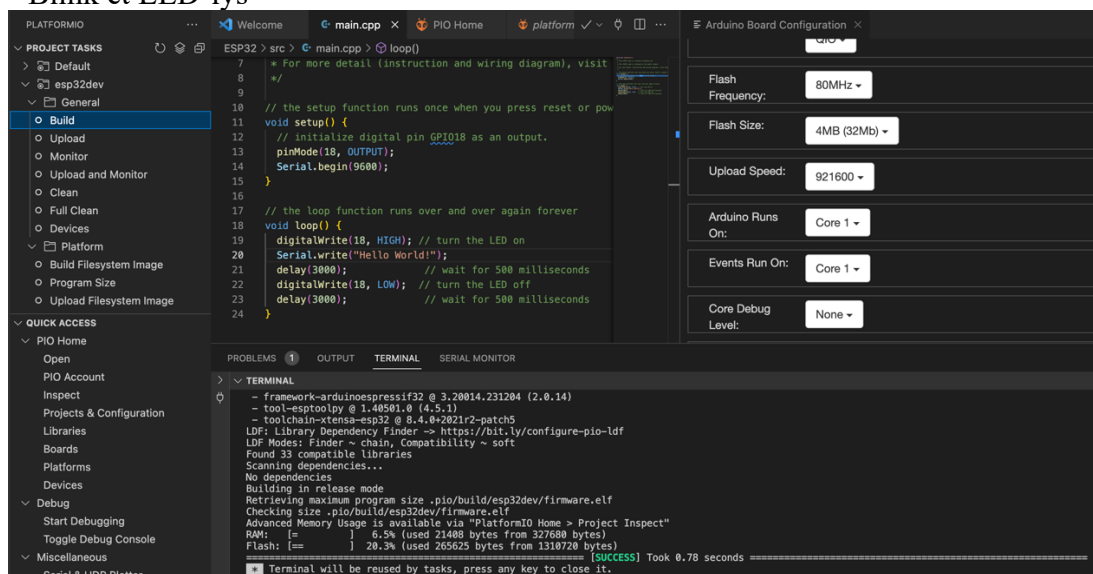


Fig 4. SUCCESS-meldingen ved opplastningen til ESP32

- Skriv til seriell-overvåker

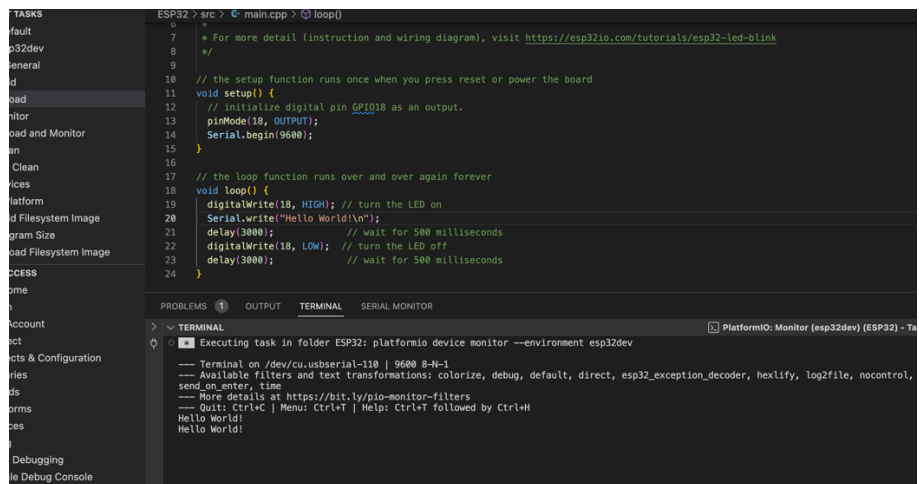


Fig 5. Seriell-overvåking

5. PWM på ESP32 (20%)

- Varier/dimm et LED-lys (det er gjort)

```

16     delay(1000);
17     Serial.print(".");
18 }
19 Serial.println("");
20 Serial.println("WiFi connected successfully");
21 Serial.print("Got IP: ");
22 Serial.println(WiFi.localIP());
23 ledcSetup(channelLED, 2000, 8);
24 ledcAttachPin(LED, channelLED);
25 ledcWrite(channelLED, 0);
26 }
27
28 void loop() {
29     for (int i = 0; i < 255; i++) {
30         ledcWrite(channelLED, i);
31         delay(10);
32     }
33     for (int i = 255; i > 0; i--) {
34         ledcWrite(channelLED, i);
35         delay(10);
36     }
37 }
38 }

```

Fig 6. Kode for variering av LED-lys med ESP32

- Bruk en buzzer (fungerte sammen med LED)

```

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

ledcSetup(channelLED, 2000, 8);
ledcSetup(channelBUZZ, 2000, 8);
ledcAttachPin(BUZZ, channelBUZZ);
ledcAttachPin(LED, channelLED);
ledcWrite(channelLED, 0);
}

void loop() {
    for (int i = 0; i < 255; i++) {
        ledcWrite(channelLED, i);
        delay(10);
    }
    ledcWriteTone(channelBUZZ, 2000);
    for (int i = 255; i > 0; i--) {
        ledcWrite(channelLED, i);
        delay(10);
    }
    ledcWriteTone(channelBUZZ, 1000);
}
}

```

Fig 7. Med buzzer i tillegg til LED (to kanaler 0-1)

- Bruk en servo (fungerte! Med tre kanaler for PWM, måtte velge forskjellige kanaler for LED og buzzer slik at det blir ingen **kanalkollisjon**)

```
#include <WiFi.h>
#include <ESP32Servo.h>
#include <HTTPClient.h>
static const int servoPin = 13;

Servo servo1;

const char *ssid = "NTNU-IOT";
const char *password = "";

const int LED = 18;
const int BUZZ = 23;
const int channelLED = 15;
const int channelBUZZ = 14;

for (int posDegrees = 0; posDegrees <= 180; posDegrees++)
{
    servo1.write(posDegrees);
    delay(20);
}

for (int posDegrees = 180; posDegrees >= 0; posDegrees--)
{
    servo1.write(posDegrees);
    delay(20);
}
```

Fig 8. Servo, LED, og Buzz tre PWM kanaler som jobber

6. Sett opp en nettside på ESP32-en som viser sensordata (20%)
 - Koble til to sensorer til ESP32-en

```
#include <Arduino.h>
#define LIGHT_SENSOR_PIN 36 // ESP32 pin GPIO36 (ADC0)
// Potentiometer is connected to GPIO 34 (Analog ADC1_CH6)
const int potPin = 34;

void setup() {
    // initialize serial communication at 9600 bits per second:
    Serial.begin(9600);
}

void loop() {
    // reads the input on analog pin (value between 0 and 4095)
    int analogValue = analogRead(LIGHT_SENSOR_PIN);
    int potValue = analogRead(potPin);
    Serial.println(potValue);

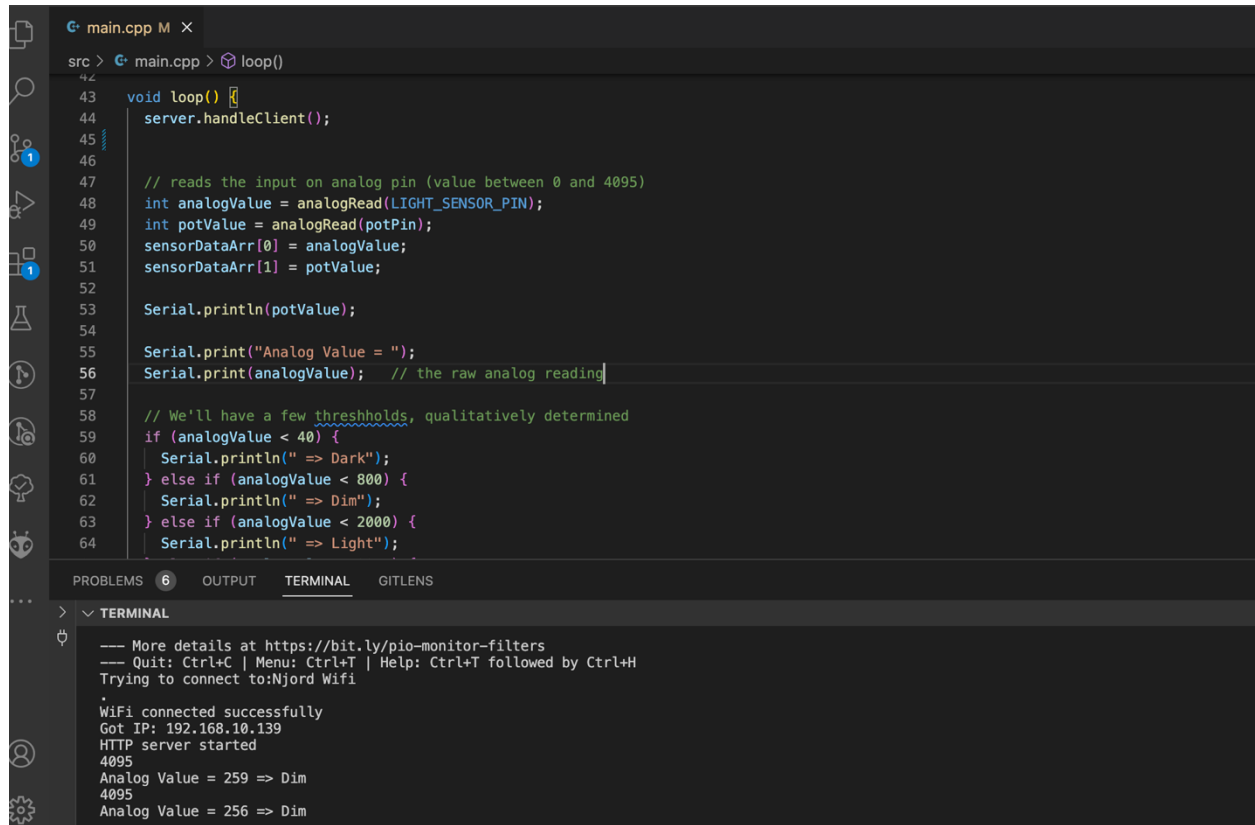
    Serial.print("Analog Value = ");
    Serial.print(analogValue); // the raw analog reading

    // We'll have a few thresholds, qualitatively determined
    if (analogValue < 40) {
        Serial.println(" => Dark");
    } else if (analogValue < 800) {
        Serial.println(" => Dim");
    } else if (analogValue < 2000) {
        Serial.println(" => Light");
    } else if (analogValue < 3200) {
        Serial.println(" => Bright");
    } else {
        Serial.println(" => Very bright");
    }

    delay(500);
}
```

Fig 9. To sensorer (ADC vs. DAC (som trenger PWM)) koblet til ESP32

- Sett opp en nettside med sensordataen



The screenshot shows an IDE with a C++ file named `main.cpp`. The code defines a `loop()` function that reads data from a light sensor and a potentiometer, then prints the values and their corresponding states (Dark, Dim, or Light) based on thresholds. The terminal output shows the device successfully connecting to WiFi, getting an IP address, and starting an HTTP server. It then displays the sensor readings: "Analog Value = 259 => Dim" and "Analog Value = 256 => Dim".

```
src > main.cpp > loop()
43 void loop() {
44     server.handleClient();
45
46
47     // reads the input on analog pin (value between 0 and 4095)
48     int analogValue = analogRead(LIGHT_SENSOR_PIN);
49     int potValue = analogRead(potPin);
50     sensorDataArr[0] = analogValue;
51     sensorDataArr[1] = potValue;
52
53     Serial.println(potValue);
54
55     Serial.print("Analog Value = ");
56     Serial.print(analogValue); // the raw analog reading
57
58     // We'll have a few thresholds, qualitatively determined
59     if (analogValue < 40) {
60         Serial.println(" => Dark");
61     } else if (analogValue < 800) {
62         Serial.println(" => Dim");
63     } else if (analogValue < 2000) {
64         Serial.println(" => Light");
65     }
```

PROBLEMS 6 OUTPUT TERMINAL GITLENS

TERMINAL

```
--- More details at https://bit.ly/pio-monitor-filters
--- Quit: Ctrl+C | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H
Trying to connect to:Njord Wifi

WiFi connected successfully
Got IP: 192.168.10.139
HTTP server started
4095
Analog Value = 259 => Dim
4095
Analog Value = 256 => Dim
```

Fig 10. ESP32 kobler til hjemme ruteren og setter opp http-serveren



Sensordata

Light Sensor: 257

Potentiometer: 1299

Fig 11. Riktig data vises hvert «refresh».


```

74 String GetHTML_Data(int sensorDataArr[]) {
75     // HTML & CSS contents which display on web server
76     String HTML = "<!DOCTYPE html>"
77     "<html>"
78     "<head>"
79     "<title>Sensordata</title>"
80     "</head>"
81     "<body>"
82     "<h1>Sensordata</h1>";
83     HTML += "<h2>Light Sensor: ";
84     HTML += sensorDataArr[0];
85     HTML += "</h2>";
86     HTML += "<h2>Potentiometer: ";
87     HTML += sensorDataArr[1];
88     HTML += "</h2>";
89     HTML += "</body>";
90     HTML += "</html>";
91     return HTML;
92 }
93
94 void handle_root() {
95     server.send(200, "text/html", GetHTML_Data(sensorDataArr));
96 }
97

```

Fig 12. Funksjon som oppdaterer HTML

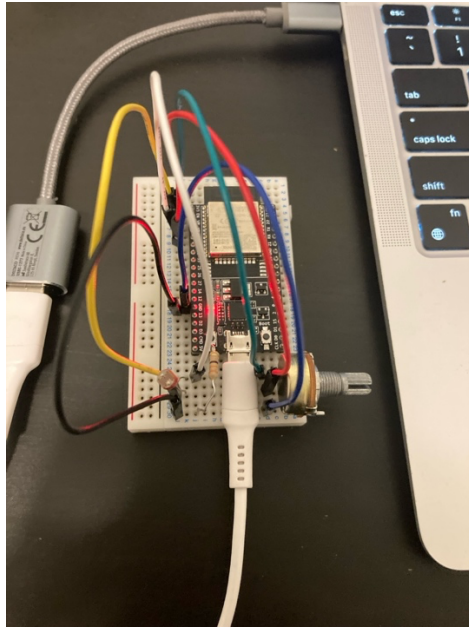


Fig 13. ESP32-koblingen med potentiometer og lyssensor