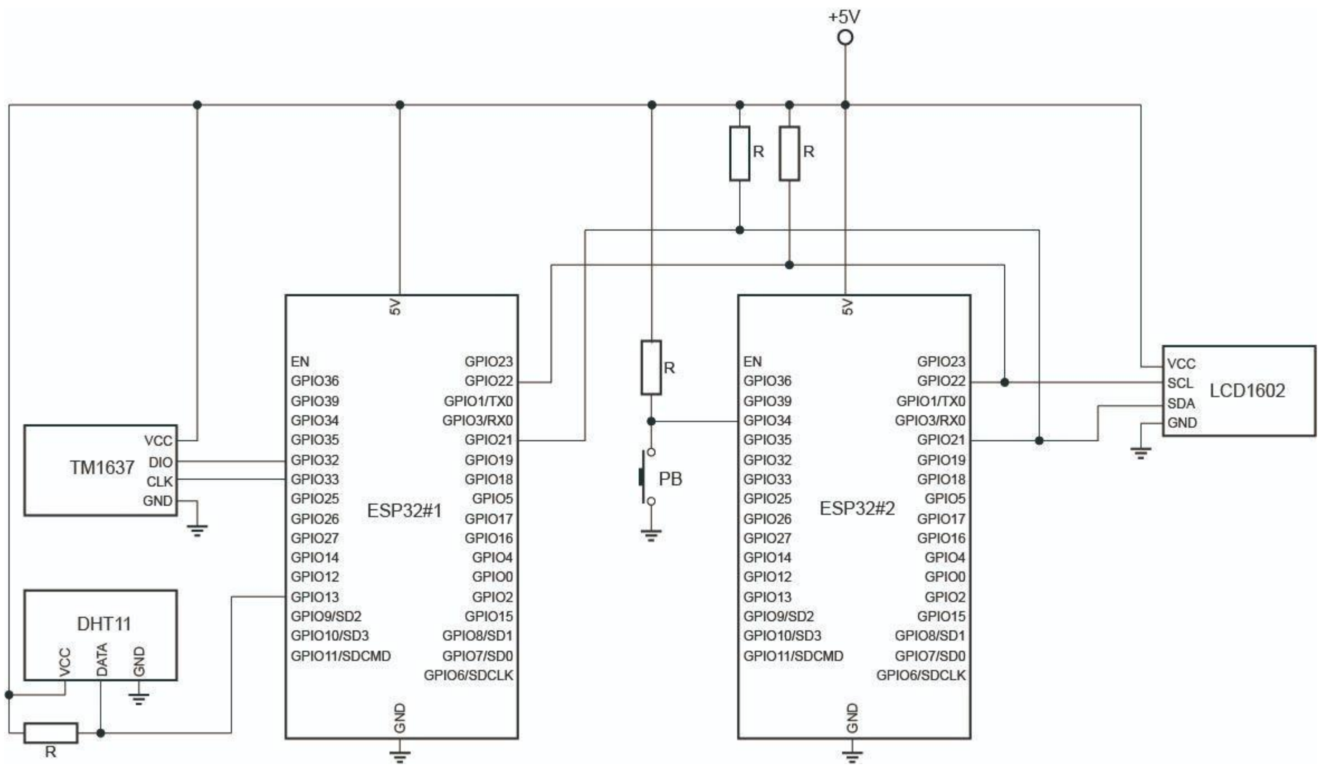
LAB ASSIGNMENT 1

1. Introduction

We used two ESP32 to communicate and display the temperature and humidity level of the surroundings. The 1st ESP32 is named "**Primary or Master ESP**" and the 2nd one is named "**Secondary or Slave ESP**". The 2nd micro controller reads the temperature and Humidity values and sends it to the 1st micro controller when requested. The 1st micro controller on receiving the value displays it on to the connected LCD. A push button is connected to the board to send a request from 1st micro controller.

2. Wiring Diagram



3. Master ESP

3.1 Libraries Included

LiquidCrystal_I2C.h

Wire.h

3.2 Address Allocations

Name	Address
Push button as BUTTON_PIN	GPIO34
SDA	GPIO21
SCL	GPIO22
LiquidCrystal_I2C	0x27

3.3 Working with LCD

- The lCD is connected at address 0x27.
- It can display upto 2 lines with 16 characters each.

```
LiquidCrystal_I2C lcd(0x27, 16, 2);
```

- The LCD is first initialised using the following:

```
lcd.begin(); // lcd.init();
lcd.backlight();
```

- To display the desired output two functions setCursor( ) and print( ) are used.

3.4 Connection to The Secondary ESP

I2c wires are connected between the primary and the secondary ESP via the Push button. The Primary ESP requests the data from destination address 4. It then receives byte values which is then further converted into integer and displayed on the LCD. A check is added to remove redundant values.

```
I2C_1.requestFrom((uint8_t)DEST_ADDR2, (uint8_t)2);

while (I2C_1.available())
        byte c = I2C_1.read();
```

3.5 Push Button

The Push Button has two states namely HIGH and Low. The Push button is configured as high or low in the beginning as it is hard to tell the current state just by looking at it. In this code, the default pinmode is set to PULLDOWN and is connected to address 34.

```
pinMode(BUTTON_PIN, INPUT_PULLDOWN);
```

Full Source code deployed on Primary ESP

```
#include <LiquidCrystal_I2C.h>
#include <Wire.h>
#define BUTTON_PIN 34

int currentState;

const u   8_t DEST_ADDR2 = 4;
const int SDA1 = 21;
const int SCL1 = 22;

LiquidCrystal_I2C lcd(0x27, 16, 2); // set the LCD address to 0x27 for a 16 chars and 2 line display
TwoWire I2C_1 = TwoWire(0);

void setup()
{
  Serial.begin(115200);
  pinMode(BUTTON_PIN, INPUT_PULLDOWN);
  I2C_1.begin(SDA1, SCL1);

  lcd.begin();                        // initialize the lcd
  lcd.backlight();
}

  void loop()
{
  currentState = digitalRead(BUTTON_PIN);
  Serial.print(currentState);

  int i = 0;
  if (currentState == LOW) {
    I2C_1.requestFrom((uint8_t)DEST_ADDR2, (uint8_t)2);
    Serial.print("request sent");
    while (I2C_1.available()) {
      byte c = I2C_1.read();
      if ((int)c != 255) {
        lcd.setCursor(3, i % 2);
        i++;
        lcd.print((int)c);
      }
    }
  }
}
```

4. Slave ESP

4.1 Libraries Included

Wire.h

DHT.h

TM1637.h

4.2 Address Allocations

Name	Address
DHT11 data pin	GPIO13
I2C bus 1 Clock	GPIO22
I2C bus 1 Data	GPIO21
I2C bus 2 Clock	GPIO33
I2C bus 2 Data	GPIO32

4.3 Reading temperature and humidity from DHT11

- The DHT11 is initialised using the following:

```
dht.begin();
```

- To read the the Temperature and Humidity the functions readHumidity( ) and read Temperature( ) are used.

```
h = dht.readHumidity();
t = dht.readTemperature();
```

- If there are some errors we print an error message. Otherwise, print the Humidity and Temperature on the serial monitor

```
if (isnan(h) || isnan(t)) {
   Serial.println("error!!");
   return;
}
Serial.printf("Humidity: %d%%\n",(int)h);
Serial.printf("Temperature: %d°C\n",(int)t);
```

4.4 Working with TM1637 module

- First, Initialize the TM1637 module:

```
tm1637.init();

tm1637.set(BRIGHT_TYPICAL);
```

- Then we can display the temperature and humidity on TM1637. The first two digits are for humidity, the third and fourth digits are for temperature.

```
numdisplay = (int)h*100 + (int)t;
tm1637.display(0,numdisplay / 1000 % 10);// 1st digit
tm1637.display(1,numdisplay / 100 % 10);// 2nd digit
tm1637.display(2,numdisplay / 10 % 10);// 3rd digit
tm1637.display(3,numdisplay % 10);// 4th digit
```

4.5 Connection to The Master ESP

- The following is the setting for the communication. When getting requests from Master, run the requestevent()

```
I2C_1.begin(MY_ADDR1,SDA1,SCL1);

I2C_1.onRequest(requestEvent);
```

- Request Event, send the Humidity and Temperature values to the Master ESP

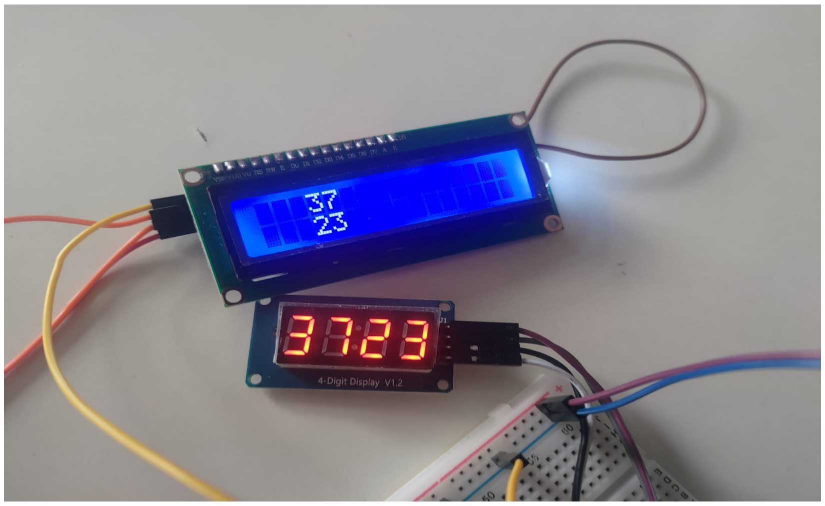
```
void requestEvent()
{
        Serial.printf("requestEvent\n");

        dataToSend1 = int(h);//Humidity
        dataToSend2 = int(t);//Temperature

        I2C_1.write(dataToSend1);
        I2C_1.write(dataToSend2);
}
```

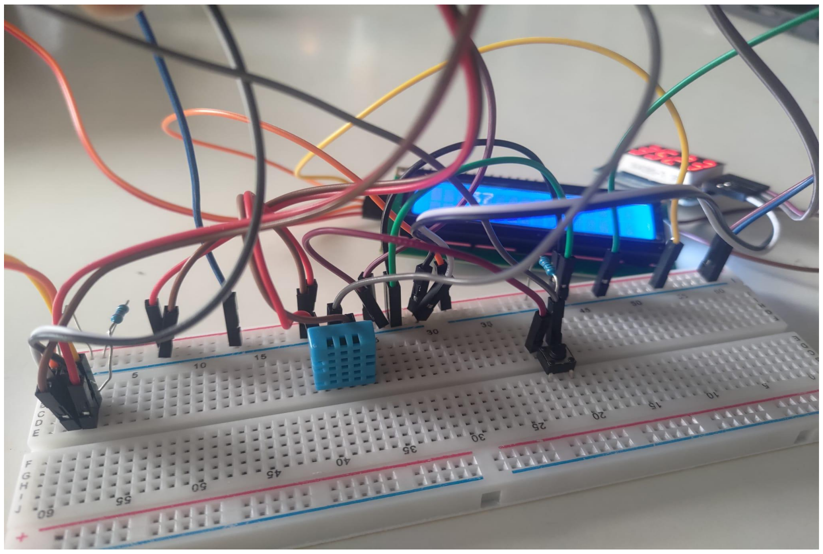
Full Source code deployed on Slave ESP

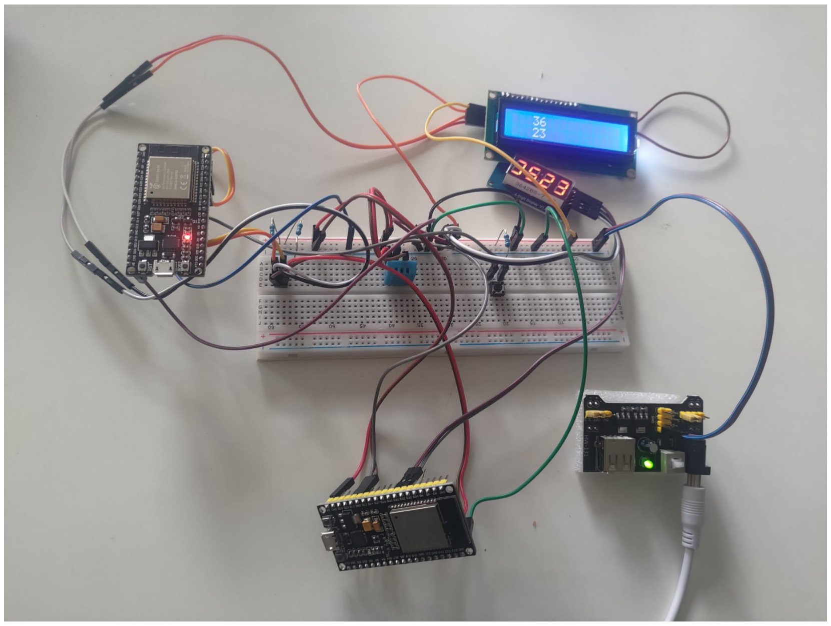
```
#include <Wire.h>
#include "DHT.h"
#include "TM1637.h"

const uint8_t MY_ADDR1 = 4;
const uint8_t dhtType = DHT11;
const uint8_t dhtPin = 13;
const uint8_t SDA1 = 21;
const uint8_t SCL1 = 22;
const uint8_t SDA2 = 32;
const uint8_t SCL2 = 33;

TwoWire I2C_1 = TwoWire(0);
byte dataToSend1 = 0;
byte dataToSend2 = 0;
DHT dht(dhtPin, dhtType);
TM1637 tm1637(SCL2,SDA2);
float h = 0;
float t = 0;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);

  I2C_1.begin(MY_ADDR1,SDA1,SCL1);
  I2C_1.onRequest(requestEvent);

  dht.begin();//initialize the DHT11 module

  tm1637.init();
  tm1637.set(BRIGHT_TYPICAL);
}
```

```
void loop() {
  // put your main code here, to run repeatedly:
  int numdisplay = 0;

  h = dht.readHumidity(); //read
  t = dht.readTemperature();
  if (isnan(h) || isnan(t)) {
    Serial.println("error!!");
    return;
  }
  Serial.printf("Humidity: %d%%\n",(int)h);
  Serial.printf("Temperature: %d°C\n",(int)t);

  numdisplay = (int)h*100 + (int)t;
  tm1637.display(0,numdisplay / 1000 % 10);// 1st digit
  tm1637.display(1,numdisplay / 100 % 10);// 2nd digit
  tm1637.display(2,numdisplay / 10 % 10);// 3rd digit
  tm1637.display(3,numdisplay % 10);// 4th digit
  delay(500);
}

void requestEvent()
{
  Serial.printf("requestEvent\n");

  dataToSend1 = int(h);//Humidity
  dataToSend2 = int(t);//Temperature

  I2C_1.write(dataToSend1);
  I2C_1.write(dataToSend2);

}
```

Working Images







Observations

- The default libraries were not behaving as per the requirements, so they were updated to the latest version.
- The contrast controller of the LCD is to be dealt with as in the first hour the code was working fine but the result could not be seen on the LCD (becoz the contrast was low).