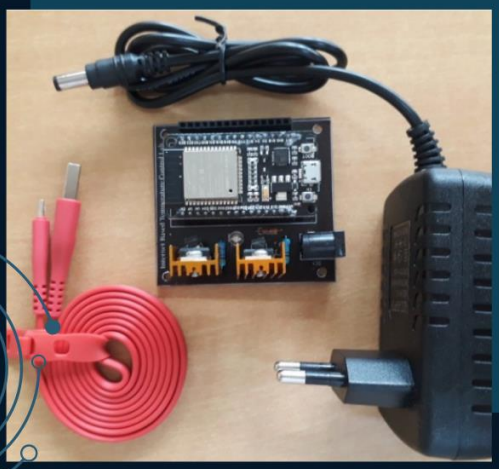
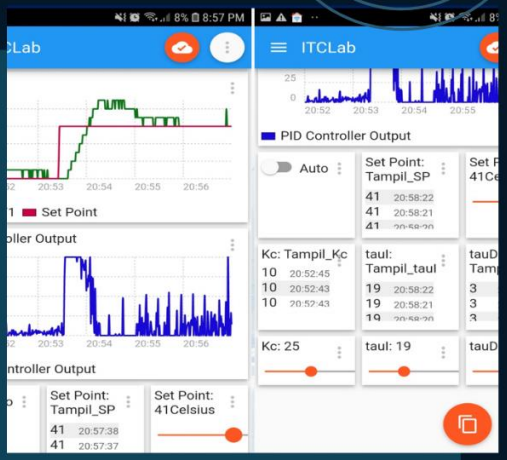
ITCLAB

INTERNET-BASED TEMPERATUE CONTROL LAB



iTCLab Kits for:
- Internet of Things (IoT)
- System Dynamics
- Control System



iTCLab Kits for:
- Machine Learning
- Arduino and Python programming
- System Modeling

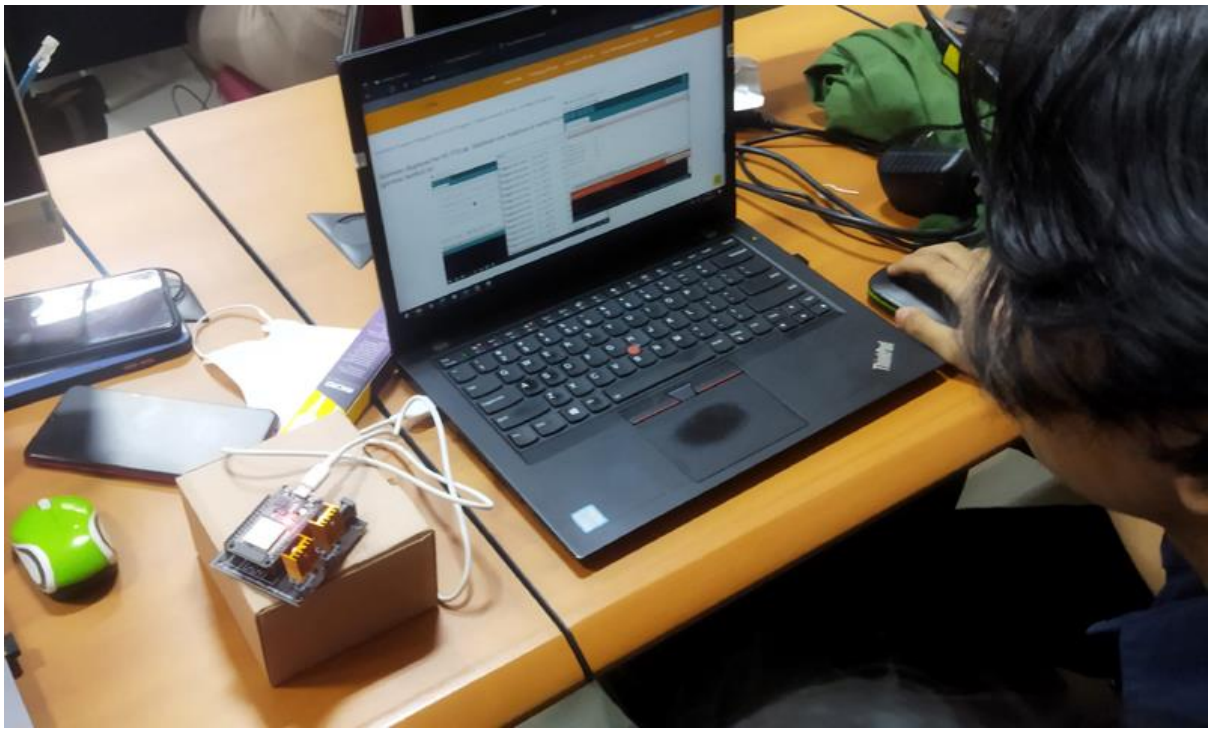
https://io-t.net/itclab

https://github.com/bsrahmat

https://shopee.co.id/product/78709625/11589970517/

https://io-t.net/itclab
https://github.com/bsrahmat
https://shopee.co.id/product/78709625/11589970517

iTCLab Testing



iTCLab - Internet-Based Temperatue Control Lab. Temperature control kit for feedback control applications with ESP32 microcontroller, LEDs, two heaters, and two temperature sensors. The heating power output is adjusted to maintain the desired temperature setpoint. Thermal energy from the heater is transferred by conduction, convection, and radiation to the temperature sensor. Heat is also transferred from the device to the environment.

iTCLab Kit:

- Inspired by BYU (Brigham Young University) TCLab Product. A private university in Provo, Utah, United States.
- Miniature Control System in Our Pocket.
- Practical IoT Learning Package.
- Introduction to IoT Systems.
- IoT Programming.
- Practice of IoT-Based Control Systems.
- Can be used to learn System Dynamics and Control Systems.
- Can be used to learn Arduino and Python Programming.
- Can be used to learn Machine Learning Programming.
- And others.

The fundamental difference between iTCLab and BYU's TCLab product is the replacement of the Arduino Uno microcontroller with the ESP32. By using the ESP32, iTCLab has the ability to connect to the Internet of Things (IoT).

iTCLab Upper Temperature Limit Description:

The upper temperature limit of the iTCLab Kit is 60 degrees Celsius. Therefore, when experimenting with this Kit, this Upper Temperature Limit must not be exceeded. Violation of this provision could cause damage (burning) to the components.

Although the upper limit is 60 degrees Celsius, it is still sufficient for experimenting with this Kit. And it is sufficient to see the performance of a control method. For example, control using Proportional Integral and Derivative (PID). Or to see the effect of tuning the PID parameters using the Machine Learning method. An illustration of the capabilities of this iTCLab Kit can be seen from the illustration of the performance of the BYU TCLab, as seen in the following simulation.

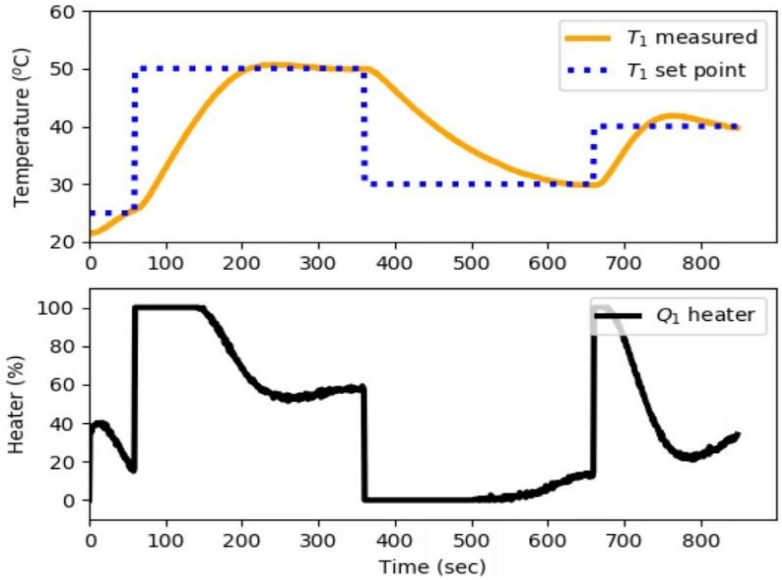


Figure 1. BYU TCLab Kit performance overview

Coding Upper Temperature Limit

It is necessary to set a limit so that the iTCLab Kit always operates in a safe area. It must not exceed the upper limit of 60 degrees Celsius. The following is an example of an Arduino program script that must be added every time you experiment with this Kit. In the Loop, it is added that if it reaches the specified upper limit (it can be lowered slightly, for example 55 degrees Celsius), then the heater must be turned off.

```
void loop() {
  // put your main code here, to run repeatedly:
  cektemp();
  if (cel > upper_temperature_limit){
    Q1off();
    ledon();
  }
  else {
    Q1on();
    ledoff();
  }
```

```
  if (cel1 > upper_temperature_limit){
    Q2off();
    ledon();
  }
  else {
    Q2on();
    ledoff();
  }
  delay (100);
}
```

iTCLab_Testing Program

iTCLab_Testing is a simple iTCLab Kit testing program. The temperature is gradually increased to the desired upper limit of 55 degrees Celsius.

Required Equipment:

- iTCLab Kit
- iTCLab_Testing.ino Program
- iTCLab_Testing.pdf Tutorial

Required Settings:

File Settings - Preferences:

The iTCLab Kit uses an ESP32 microcontroller. Please copy and paste the following address in the File - Preferences section:

Address: https://dl.espressif.com/dl/package_esp32_index.json

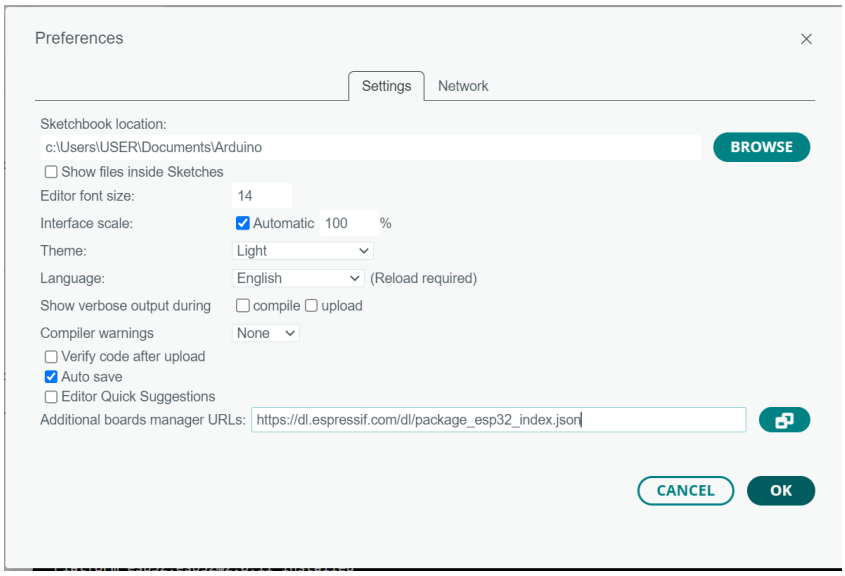


Figure 2. File Settings – Preferences

Board Settings for DOIT ESP32 DEVKIT V1

ESP32 Microcontroller Board Settings, in the Menu Tools - Board - Boards Manager. Please select DOIT ESP32 DEVKIT V1.

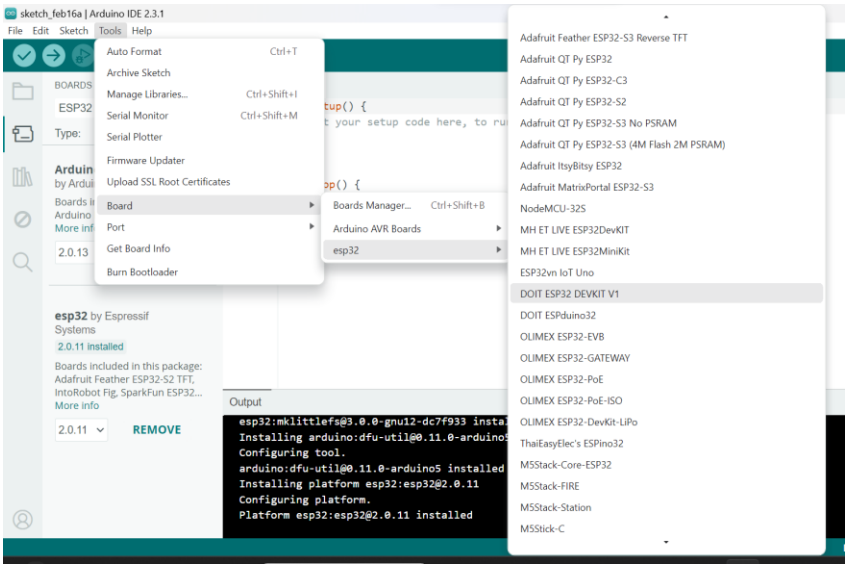


Figure 3. Selecting the DOIT ESP32 DEVKIT V1 Board

iTCLab Port Selection:

Please select the port that corresponds to the iTCLab Kit when it is connected to the computer (laptop).

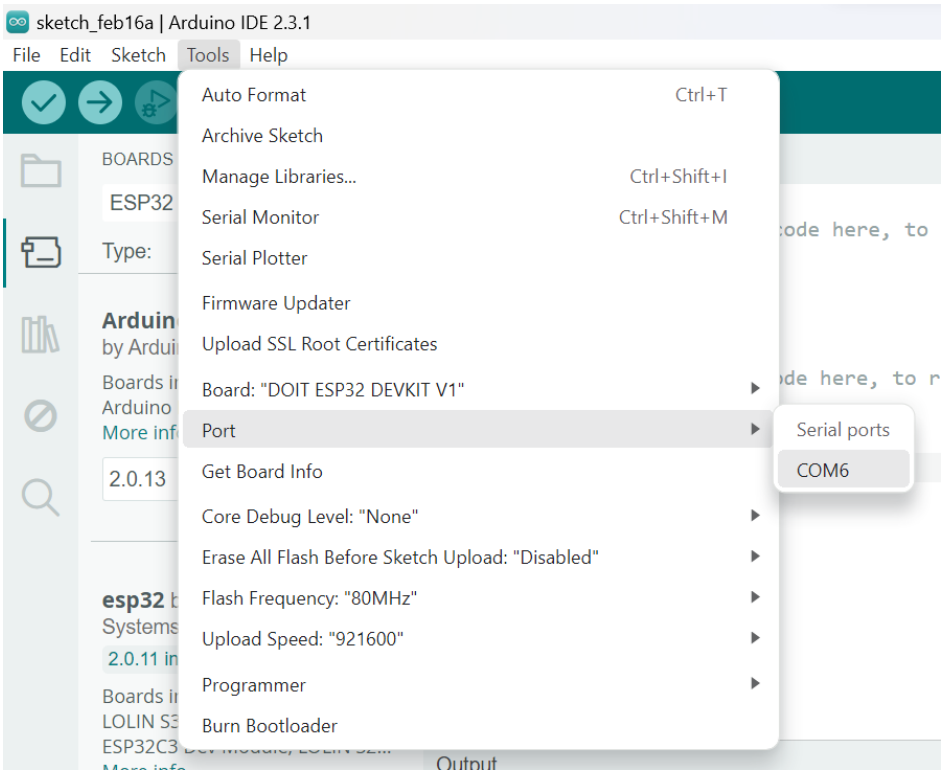


Figure 4. iTCLab Port Selection

Upload speed setting

Please select the upload speed, at 115200.

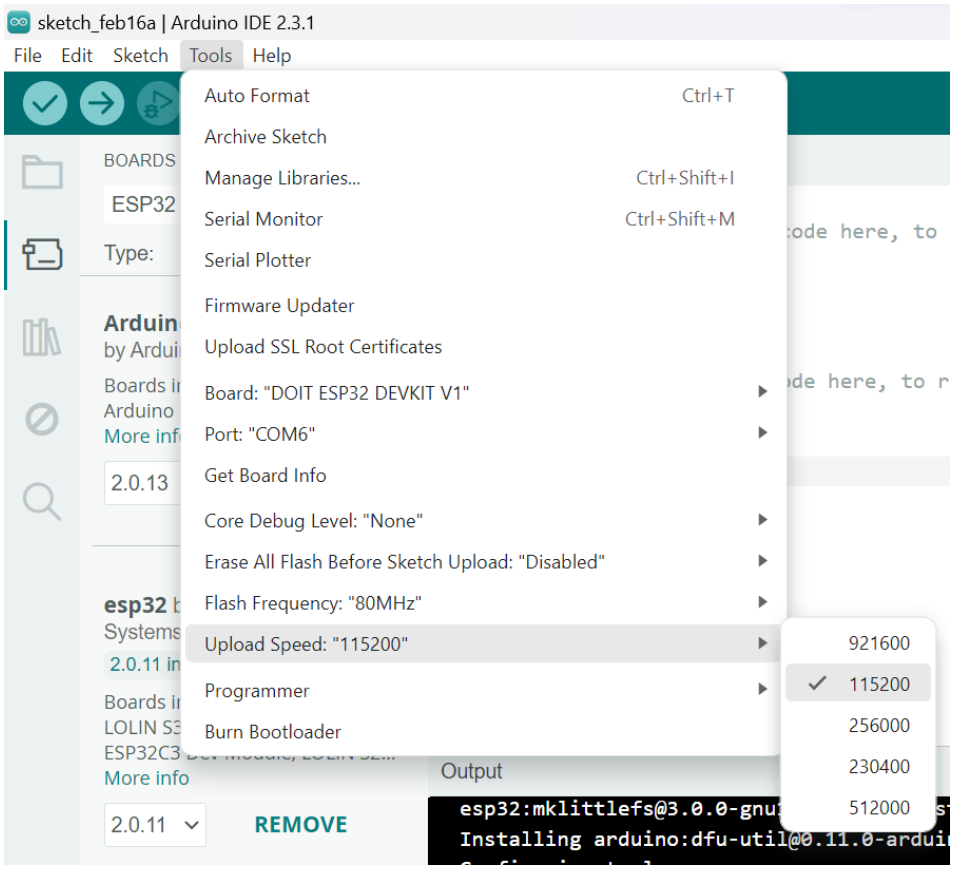


Figure 5. Upload speed setting

Here is the iTCLab_Testing.ino coding script:

Program : iTCLab_Testing

```
/************************************
 * Program : iTCLab_Testing
 * By      : io-t.net Team
 * ***********************************/

#include <Arduino.h>

// constants
const int baud = 115200;       // serial baud rate

// pin numbers corresponding to signals on the iTCLab Shield
const int pinT1   = 34;         // T1
const int pinT2   = 35;         // T2
const int pinQ1   = 32;         // Q1
```

```
const int pinQ2   = 33;          // Q2
const int pinLED  = 26;          // LED

// setting PWM properties
const int freq = 5000; //5000
const int ledChannel = 0;
const int Q1Channel = 1;
const int Q2Channel = 2;
const int resolutionLedChannel = 8; //Resolution 8, 10, 12, 15
const int resolutionQ1Channel = 8; //Resolution 8, 10, 12, 15
const int resolutionQ2Channel = 8; //Resolution 8, 10, 12, 15

float cel, cel1, degC, degC1;
const float upper_temperature_limit = 55;

// global variables
float Q1 = 0;                  // value written to Q1 pin
float Q2 = 0;                  // value written to Q2 pin
int iwrite_max = 255;           // integer value for writing
int iwrite_min = 0;             // integer value for writing

void setup() {
  // put your setup code here, to run once:
  Serial.begin(baud);
  while (!Serial) {
    ; // wait for serial port to connect.
  }

  // configure pinQ1 PWM functionalitites
  ledcSetup(Q1Channel, freq, resolutionQ1Channel);

  // attach the channel to the pinQ1 to be controlled
  ledcAttachPin(pinQ1, Q1Channel);

  // configure pinQ2 PWM functionalitites
  ledcSetup(Q2Channel, freq, resolutionQ2Channel);

  // attach the channel to the pinQ2 to be controlled
  ledcAttachPin(pinQ2, Q2Channel);

  // configure pinLED PWM functionalitites
  ledcSetup(ledChannel, freq, resolutionLedChannel);

  // attach the channel to the pinLED to be controlled
  ledcAttachPin(pinLED, ledChannel);

  ledcWrite(Q1Channel,0);
  ledcWrite(Q2Channel,0);
```

```
  ledcWrite(ledChannel,0);
}

void Q1on(){
    ledcWrite(Q1Channel,iwrite_max/255*100);
    //Q1 = iwrite_max/255*100;
    //Serial.println(Q1);
}

void Q1off(){
    ledcWrite(Q1Channel,iwrite_min/255*100);
    //Q1 = iwrite_min/255*100;
    //Serial.println(Q1);
}

void Q2on(){
    ledcWrite(Q2Channel,iwrite_max/255*100);
    //Q2 = iwrite_max/255*100;
    //Serial.println(Q2);
}

void Q2off(){
    ledcWrite(Q2Channel,iwrite_min/255*100);
    //Q2 = iwrite_min/255*100;
    //Serial.println(Q2);
}

void ledon(){
    ledcWrite(ledChannel,iwrite_max);
}

void ledoff(){
    ledcWrite(ledChannel,iwrite_min);
}

void cektemp(){
  degC = analogRead(pinT1) * 0.322265625 ;    // use for 3.3v AREF
  cel = degC/10;
  degC1 = analogRead(pinT2) * 0.322265625 ;    // use for 3.3v AREF
  cel1 = degC1/10;

  Serial.print("Temperature: ");
  Serial.print(cel);   // print the temperature T1 in Celsius
  Serial.print("°C");
  Serial.print("  ~  "); // separator between Celsius and Fahrenheit
  Serial.print(cel1);   // print the temperature T2 in Celsius
  Serial.println("°C");
}
```

```
void loop() {
  // put your main code here, to run repeatedly:
  cektemp();
  if (cel > upper_temperature_limit){
    Q1off();
    ledon();
  }
  else {
    Q1on();
    ledoff();
  }
  if (cel1 > upper_temperature_limit){
    Q2off();
    ledon();
  }
  else {
    Q2on();
    ledoff();
  }
  delay (100);
}
```

Please upload the code above to the iTCLab Kit, wait until the process is finished. After it has been successfully uploaded. Please check the results on the Serial Monitor.

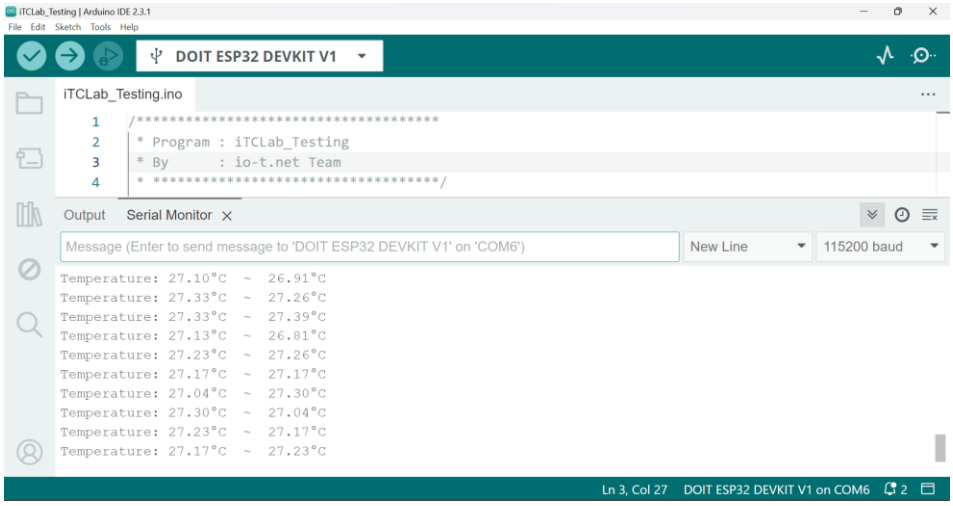


Figure 6. iTCLab Temperature Reading Results