Course of Medical Design (BM017IU)

Semester 1, 2024-2025

Instructor: Dr. Tran Le Giang

LAB 3 HANDOUT

1. Objectives

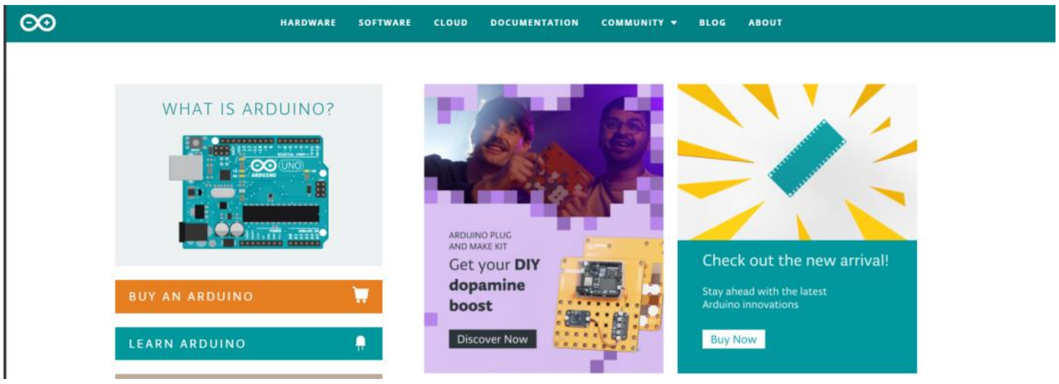
- Introduction to installation of IDE for ESP32
- Getting Started with GitHub and Git
- Introduction to State Machine

2. Lab contents

2.1.Introduction to installation of IDE for ESP32 (*Arduino IDE and PlatformIO*)

To install the Arduino IDE on Windows, you can follow these steps:

1. Go to **Arduino.cc**

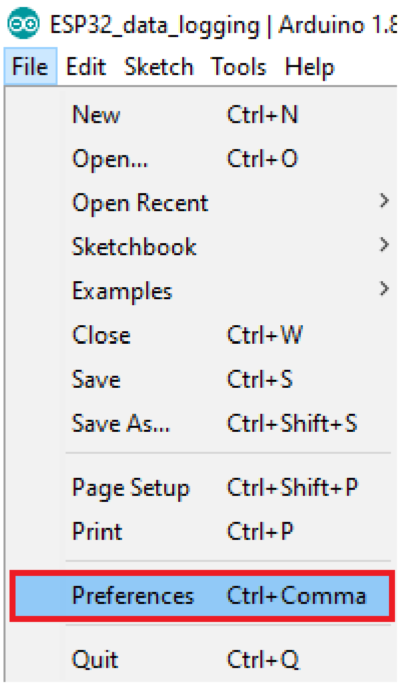

HARDWARE
SOFTWARE
CLOUD
DOCUMENTATION
COMMUNITY
BLOG
ABOUT
WHAT IS ARDUINO?
BUY AN ARDUINO
LEARN ARDUINO
ARDUINO PLUG AND MAKE KIT
Get your DIY dopamine boost
Discover Now
Check out the new arrival!
Stay ahead with the latest Arduino innovations
Buy Now

2. Hover over Software and click Downloads
3. Click the **Download Link** for the Windows installer
4. Open the downloaded file
5. Agree to the license agreement
6. Select what to install
7. Choose the installation path
8. Click Install
9. Wait for the installation to finish
10. Click Close to end the setup wizard

Installing ESP32 Add-on in Arduino IDE

To install the ESP32 board in your Arduino IDE, follow these next instructions:

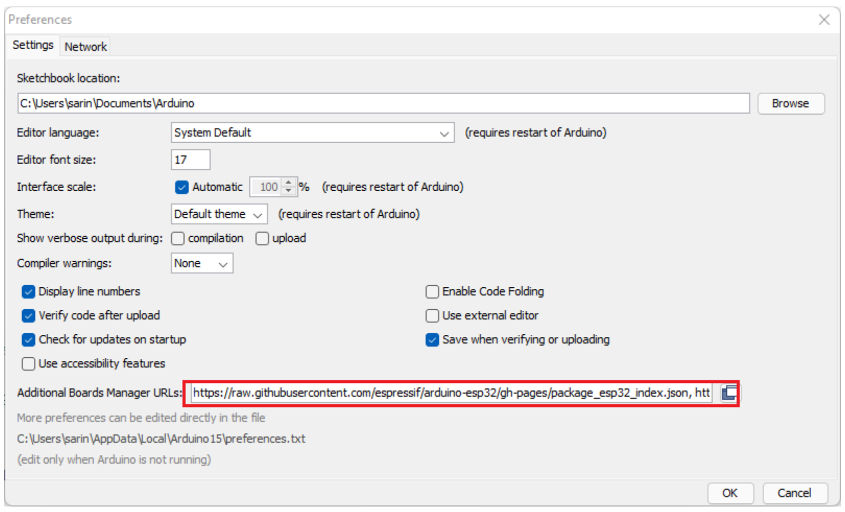
1. In your Arduino IDE, go to **File**> **Preferences**



2. Enter the following into the "Additional Board Manager URLs" field:

https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json

Then, click the "OK" button:


Preferences

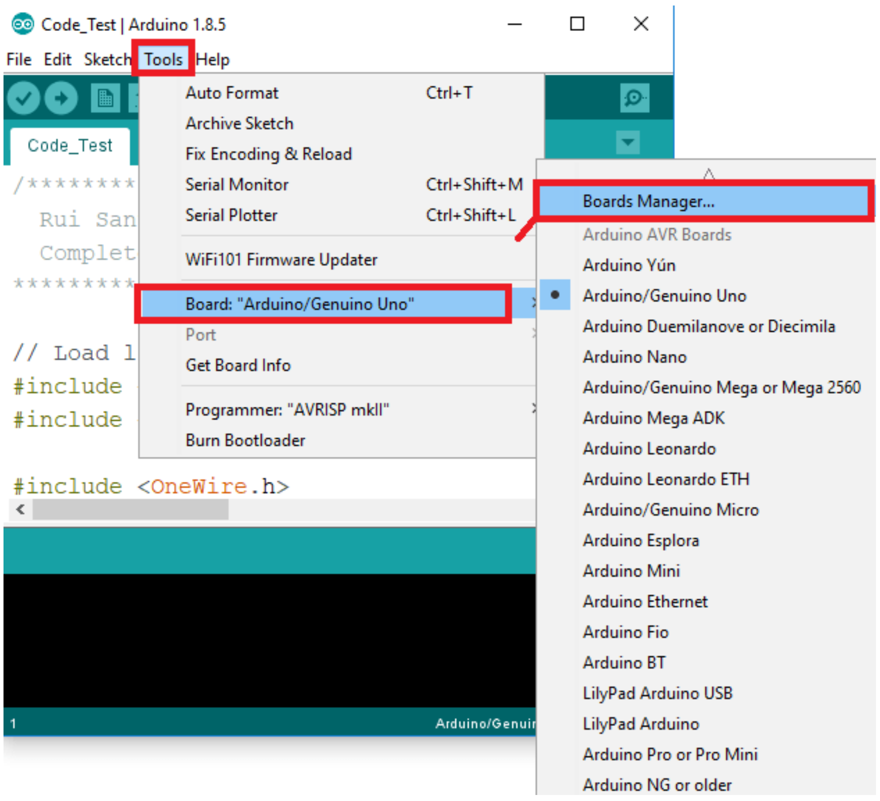
Settings Network

Sketchbook location:
C:\Users\sarin\Documents\Arduino
Editor language: System Default (requires restart of Arduino)
Editor font size: 17
Interface scale: ✓ Automatic 100 % (requires restart of Arduino)
Theme: Default theme (requires restart of Arduino)
Show verbose output during: ☐ compilation ☐ upload
Compiler warnings: None
✓ Display line numbers ☐ Enable Code Folding
✓ Verify code after upload ☐ Use external editor
✓ Check for updates on startup ✓ Save when verifying or uploading
☐ Use accessibility features
Additional Boards Manager URLs: https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json, htt
More preferences can be edited directly in the file
C:\Users\sarin\AppData\Local\Arduino15\preferences.txt
(edit only when Arduino is not running)

Note: if you already have the ESP8266 boards URL, you can separate the URLs with a comma as follows:

https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json, http://arduino.esp8266.com/stable/package_esp8266com_index.json

3. Open the Boards Manager. Go to **Tools** > **Board** > **Boards Manager…**

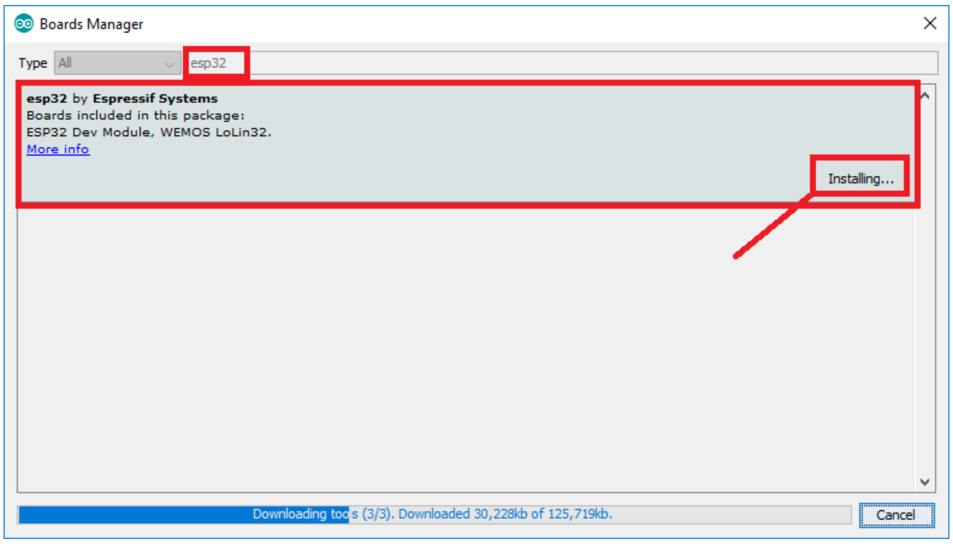


```
/********
  Rui San
  Complet
*********

// Load l
#include
#include

#include <OneWire.h>
```

4. Search for **ESP32** and press install button for the “**ESP32 by Espressif Systems**“:


Boards Manager

Type All esp32

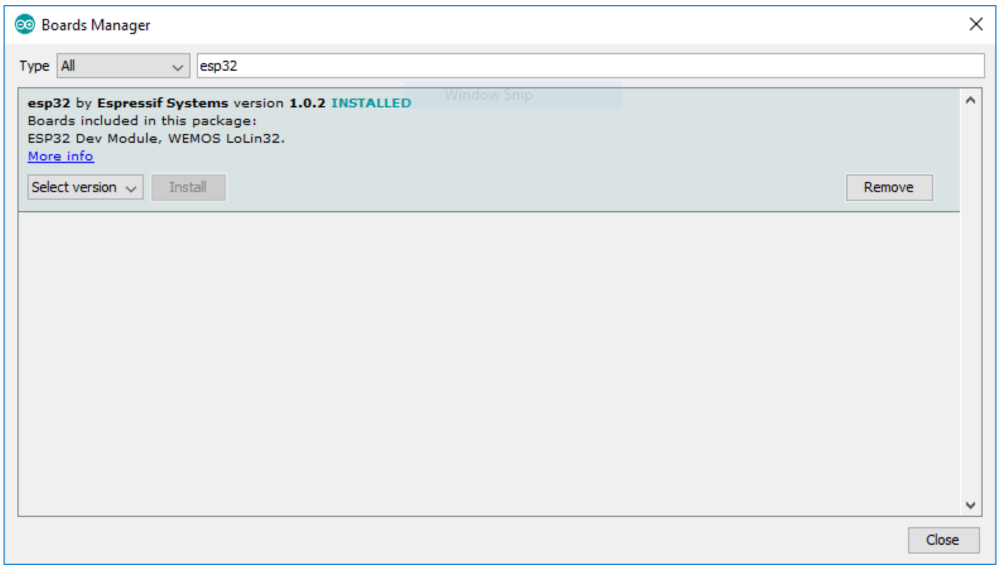
esp32 by Espressif Systems
Boards included in this package:
ESP32 Dev Module, WEMOS LoLin32.
More info

Installing...

Downloading tools (3/3). Downloaded 30,228kb of 125,719kb.

Cancel

5. That's it. It should be installed after a few seconds.


Boards Manager

Type All esp32

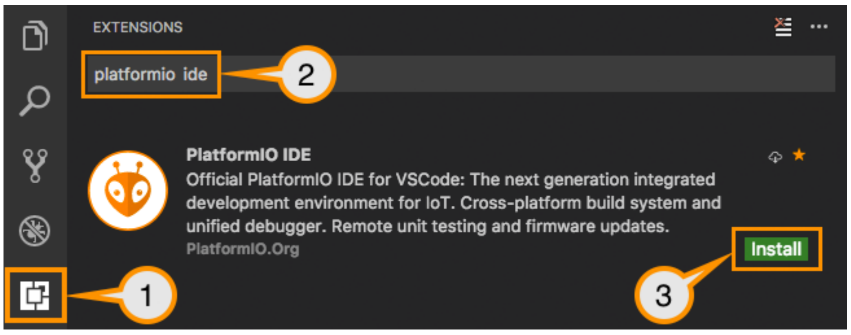
esp32 by Espressif Systems version 1.0.2 INSTALLED
Boards included in this package:
ESP32 Dev Module, WEMOS LoLin32.
More info

Select version Install Remove

Close

Installing PlatformIO on Visual Studio Code

https://docs.platformio.org/en/latest/integration/ide/vscode.html

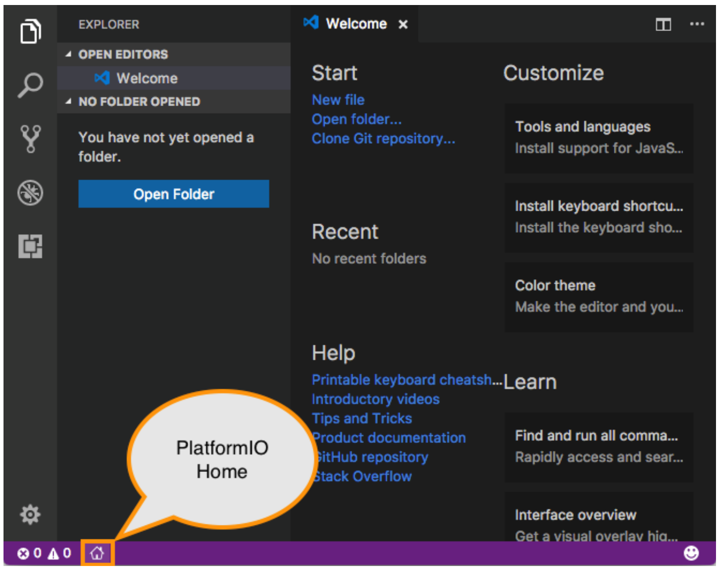

EXTENSIONS

platformio ide

PlatformIO IDE
Official PlatformIO IDE for VSCode: The next generation integrated development environment for IoT. Cross-platform build system and unified debugger. Remote unit testing and firmware updates.
PlatformIO.Org

1. Download and install official Microsoft Visual Studio Code. PlatformIO IDE is built on top of it
2. **Open** VSCode Package Manager
3. **Search** for the official platformio ide extension
4. **Install** PlatformIO IDE.

Setting Up the Project



You have not yet opened a folder.

No recent folders

Install support for JavaS...

Install the keyboard sho...

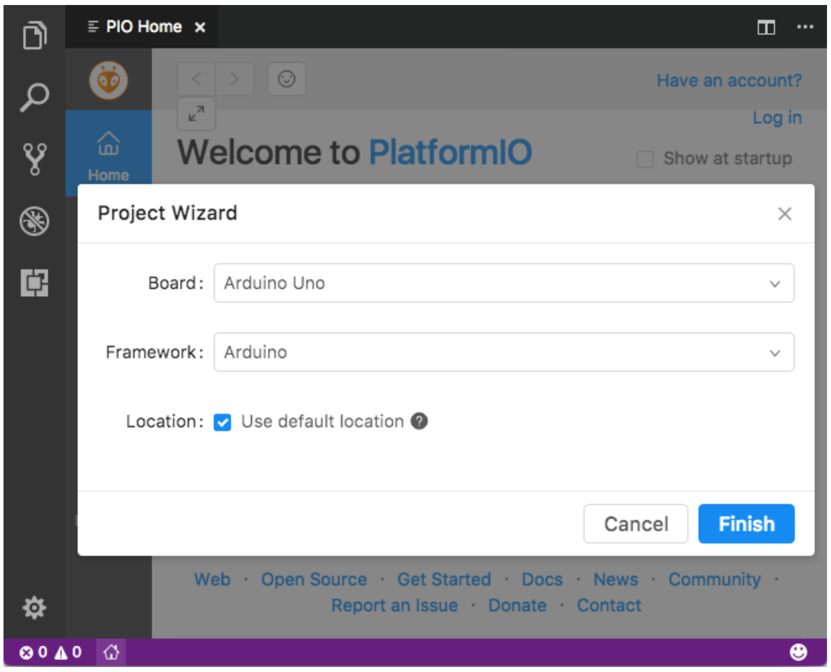
Make the editor and you...

Rapidly access and sear...

Get a visual overlay hig...

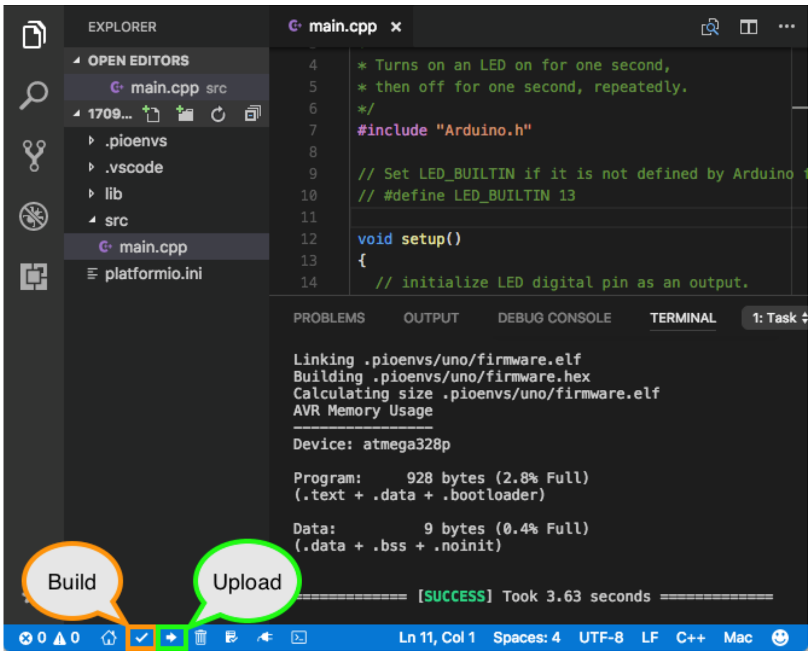
Click on "PlatformIO Home" button on the bottom PlatformIO Toolbar

Click on "New Project", select a board and create new PlatformIO Project



Board: Arduino Uno
Framework: Arduino
Location: ✓ Use default location

Open main.cpp **file from** src **folde**r **and replace its contents with your code:**



```
 * Turns on an LED on for one second,
 * then off for one second, repeatedly.
 */
#include "Arduino.h"

// Set LED_BUILTIN if it is not defined by Arduino
// #define LED_BUILTIN 13

void setup()
{
  // initialize LED digital pin as an output.
```

```
Linking .pioenvs/uno/firmware.elf
Building .pioenvs/uno/firmware.hex
Calculating size .pioenvs/uno/firmware.elf
AVR Memory Usage
----------------
Device: atmega328p

Program:     928 bytes (2.8% Full)
(.text + .data + .bootloader)

Data:          9 bytes (0.4% Full)
(.data + .bss + .noinit)

============ [SUCCESS] Took 3.63 seconds ============
```

Build your project with ctrl+alt+b **hotkey (see all Key Bindings in “User Guide” section below) or using** “Build” **button on the PlatformIO Toolbar**

2.2.Getting Started with GitHub and Git

This guide will walk you through the basics of using Git and GitHub for version control and collaboration.

2.2.1. Setting Up Your Environment

Install Git: Download and install the latest version of Git from the official website (https://git-scm.com/downloads).

Create a GitHub Account: If you don't already have one, sign up for a free GitHub account at https://github.com/.

2.2.2. Understanding Git Basics

Git is a distributed version control system that tracks changes to your files over time. Here are some key concepts:

2.2.3. Initializing a Git Repository

Create a new repository:

On GitHub: Click the "+" button in the top right corner of the GitHub website and select "New repository."

Locally: Open your terminal or command prompt, navigate to your project directory, and run **git init**.

Connect your local repository to GitHub:

Copy the remote repository URL from GitHub.

In your terminal, run ***git remote add origin <remote_repository_url>.***

2.2.4. Making Changes and Tracking Them with Git

1. **Stage changes:** After making changes to your files, use git add <filename> **or git add .** (to stage all changes) to stage them for commitment.
2. **Commit changes:** Use git commit -m "Your commit message" to create a commit with a descriptive message.

2.2.5. Working with Branches

3. **Create a new branch:** Use ***git branch <branch_name>.***
4. **Switch to a different branch:** Use ***git checkout <branch_name>.***
5. **Merge branches:** Use ***git merge <branch_name>*** to merge the specified branch into your current branch.

2.2.6. Collaborating with GitHub

6. **Push your changes:** Use git push origin <branch_name> to push your local commits to the remote repository.
7. **Pull changes from others:** Use git pull origin <branch_name> to download changes from the remote repository to your local copy.
8. **Fork a repository:** Create your own copy of someone else's repository to experiment with changes or contribute to their project.
9. **Create a pull request:** Propose changes to someone else's repository by submitting a pull request.

2.2.7. Useful Git Commands

git status: Shows the status of your working directory and staging area.
git log: Displays the commit history.
git diff: Shows the differences between your working directory and the last commit.
git checkout -- <filename>: Discards changes to a specific file.
git reset HEAD <filename>: Unstages a file.

GitHub Help: https://help.github.com/

Git Documentation: https://git-scm.com/doc

Pro Git Book: https://git-scm.com/book/en/v2

2.3.Introduction to State Machine

controlLED.cpp

```
#include "controlLED.h"


//extern unsigned char stateOfMachine;
//uint8_t stateOfLED;
/*************************************************************************************
* Function Name  : setupModeLED
* Description    :
* Input          : time delay to change the state of LED
* Output      : None
* Return      : None
* Attention   : Modify last time 18/09/2023
****************************************************************************
***********/
void setupModeLED(void)
{
  // Setup
  pinMode(LED1_PIN,OUTPUT);
  pinMode(LED2_PIN,OUTPUT);
  pinMode(LED3_PIN,OUTPUT);

  pinMode(BUTTON1_PIN,INPUT);
}
/***************************************************************************
***********
* Function Name  : blinkingLED
* Description    :
* Input         : time delay to change the state of LED
* Output        : None
* Return        : None
* Attention     : Modify last time 18/09/2023
****************************************************************************
***********/
void blinkingLED(unsigned int timeDelay)
{
  static uint16_t cnt=0, stateOfLED;
  switch(stateOfLED)
  {
    case LED_STATE1: // this state make LED on
      digitalWrite(LED1_PIN,LOW);
```

```
        digitalWrite(LED2_PIN,LOW);
        digitalWrite(LED3_PIN,LOW);
        delay(1);
        cnt++;
        if(cnt==timeDelay)
        {
          cnt=0;
          stateOfLED = LED_STATE2;
        }
      break;
      case LED_STATE2: // this state make LED off
        digitalWrite(LED1_PIN,HIGH);
        digitalWrite(LED2_PIN,HIGH);
        digitalWrite(LED3_PIN,HIGH);
        delay(1);
        cnt++;
        if(cnt==timeDelay)
        {
          cnt=0;
          stateOfLED = LED_STATE1;
        }
      break;
    }
  }
  /*************************************************************************************
  * Function Name  : alternateLED
  * Description    :
  * Input          : time delay to change the state of LED
  * Output         : None
  * Return         : None
```

```
      * Attention    : Modify last time 18/09/2023
      ***********************************************************************
**********/
      void alternateLED(unsigned int timeDelay)
      {
        static uint16_t cnt=0, stateOfLED;
        switch(stateOfLED)
        {
          case LED_STATE1:
            digitalWrite(LED1_PIN,LOW);
            digitalWrite(LED2_PIN,HIGH);
            digitalWrite(LED3_PIN,LOW);
            delay(1);
            cnt++;
            if(cnt==timeDelay)
            {
              cnt=0;
              stateOfLED = LED_STATE2;
            }
          break;
          case LED_STATE2:
            digitalWrite(LED1_PIN,HIGH);
            digitalWrite(LED2_PIN,LOW);
            digitalWrite(LED3_PIN,HIGH);
            delay(1);
            cnt++;
            if(cnt==timeDelay)
            {
              cnt=0;
              stateOfLED = LED_STATE1;
            }
          break;
        }
      }
```

controlLED.h

```
/*
 *  controlLED.h
 *  Purpose:
 *  Created on: Sept 18, 2023
 *      Author: Tran Ngoc Viet
 *  Note(s): Belong to School of Biomedical Engineering - International University
                              - Vietnam National University, HCMC
 */
#ifndef CONTROL_H__
#define CONTROL_H__
#include <Arduino.h>
// define macro
//#define LED1_PIN 18 //static int ledPin = 18;
//#define LED2_PIN 19
//#define LED3_PIN 21
//// function prototype
//void setupModeLED(void);
//void blinkLED(unsigned int t);
//void alternateLED(unsigned int t);
// Define macro
#define LED1_PIN 18
#define LED2_PIN 19
#define LED3_PIN 21
#define BUTTON1_PIN 26

#define CHECK_BUTTON 0
```

```
#define BLINK_LED 1
#define ALTERNATE_LED 2

#define LED_STATE1 0
#define LED_STATE2 1
// Function prototype
void delay_ms(uint16_t timeDelay);
void setupModeLED(void);
void alternateLED(unsigned int timeDelay);
void blinkingLED(unsigned int timeDelay);

#endif
```

Main_code

```
 */
#include "controlLED.h"
// declare globle variables
unsigned char stateOfMachine;
void setup()
{
  Serial.begin(115200);
  setupModeLED();
}

void loop()
{
//  blinkingLED(500);
//  alternateLED(500);
  if(digitalRead(BUTTON1_PIN)==0)
  {
   while(!digitalRead(BUTTON1_PIN));
   if(stateOfMachine==BLINK_LED)
   {
     stateOfMachine=ALTERNATE_LED;
      Serial.println("ALTERNATE_LED");
   }
   else
   {
     Serial.println("BLINK_LED");
     stateOfMachine=BLINK_LED;
   }
 }
 switch(stateOfMachine)
 {
   case BLINK_LED:
   blinkingLED(200);
   break;
   case ALTERNATE_LED:
   alternateLED(200);
   break;
 }
}
```

3. Lab activities & work submission

- Run the current code using state machine for LED
- Create your own repository on GitHub
- Upload and push file to GitHub using GitHub Desktop

4. Manual Information

This manual is developed by:

Dr. Tran Le Giang (Instructor)

Do Vy Ngoc (Teaching assistant)

Ta Minh Tri (Teaching assistant)

Nguyen Nhat Minh (Teaching assistant)

Last updated: Semester 1, 2024-2025