

# EECS 388 Lab #1

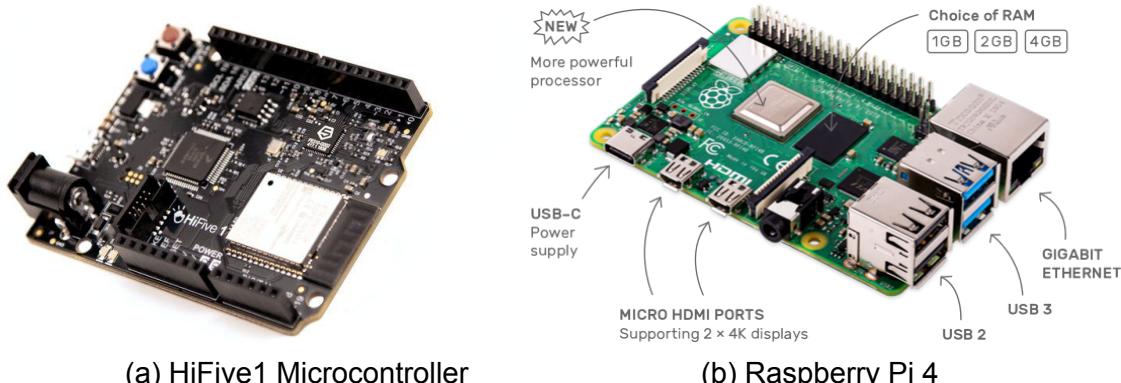
## Introduction

Welcome to EECS388 lab.

In this semester, you will develop parts of [a self-driving car](#) (watch the video in the link) using small embedded computers, sensors, and actuators. In the process, you will learn the fundamental concepts and practical skills to design and implement an embedded system.

## Hardware Platforms

We will use two embedded single board computer (SBC) platforms, shown below.



The first is an Arduino compatible SBC featuring a RISC-V architecture microcontroller called HiFive1, which will be responsible for basic control and safety of the car, and the second is a Raspberry Pi 4, which will be responsible for vision-based steering using deep learning. In the first half of the semester, we will use the HiFive1 while in the second half of the semester, we will use both platforms.

More detailed technical specs can be found in the following links. We will provide additional details of the hardware platforms when necessary for the labs in the future.

SiFive HiFive 1 rev b: <https://www.sifive.com/boards/hifive1-rev-b>

Raspberry Pi 4: <https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/>

## Part 1: Setup development environment

For software development on the microcontroller, we will use Visual Studio Code (VSCode) and PlatformIO IDE combination. VSCode is already installed on your computer but you will need to install the PlatformIO IDE and other extensions.

(Note that the following installation instructions are based on the PlatformIO IDE for VSCode documentation at: <https://docs.platformio.org/en/latest/ide/vscode.html#installation>)

### Task 1.1: Take a look at Visual Studio Code

Launch the Visual Studio Code program from the command line as follows.

```
$ code
```

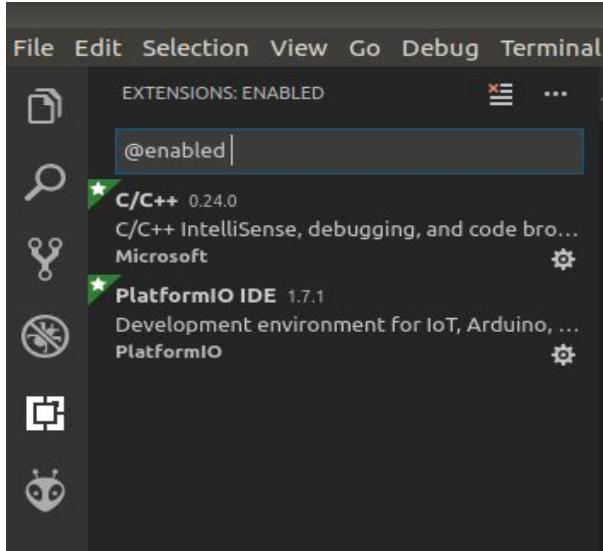
### Task 1.2: Install PlatformIO extensions

Next, Install the PlatformIO IDE extension for VSCode as follows.

1. **Open** VSCode Package Manager
2. **Search** for official “platformio ide” [extension](#)
3. **Install** PlatformIO IDE

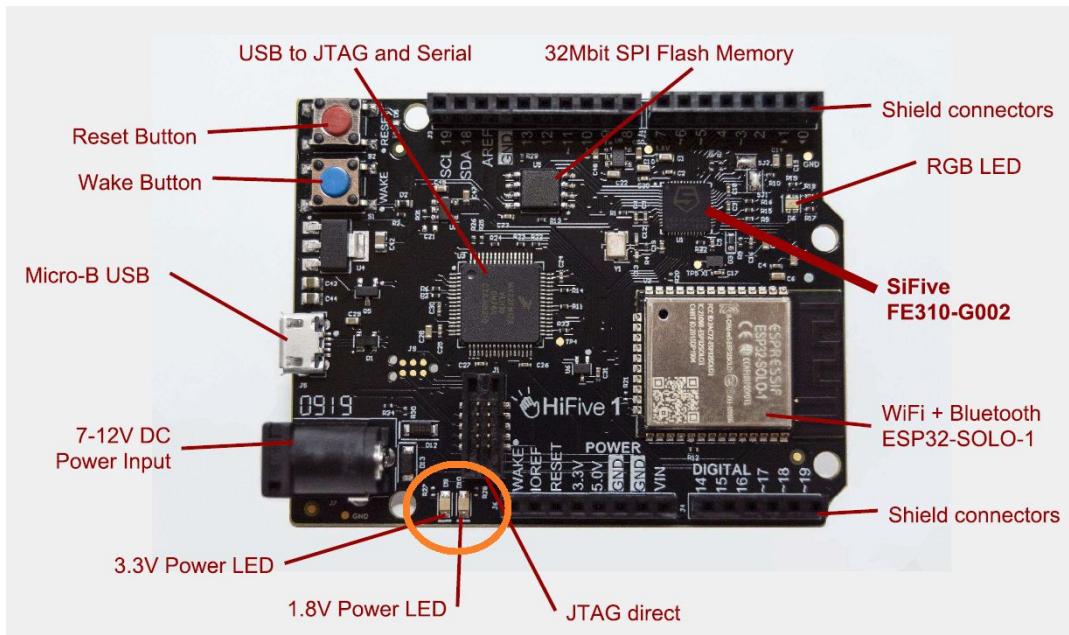


After the installation is completed, check if you have both ‘PlatformIO IDE’ and ‘C/C++’ extensions installed as follows. Optionally, installing ‘vscode-icons’ and ‘vscode-pdf’ extensions are also recommended.



### Task 1.3. Connect the board to your PC

Next, connect your board to one of your PC's USB ports. Once the board is connected to the PC, two yellow power LEDs (3.3V and 1.8V) should be turned on. (See the figure below).



## Part 2: Run your first program on the HiFive1 board.

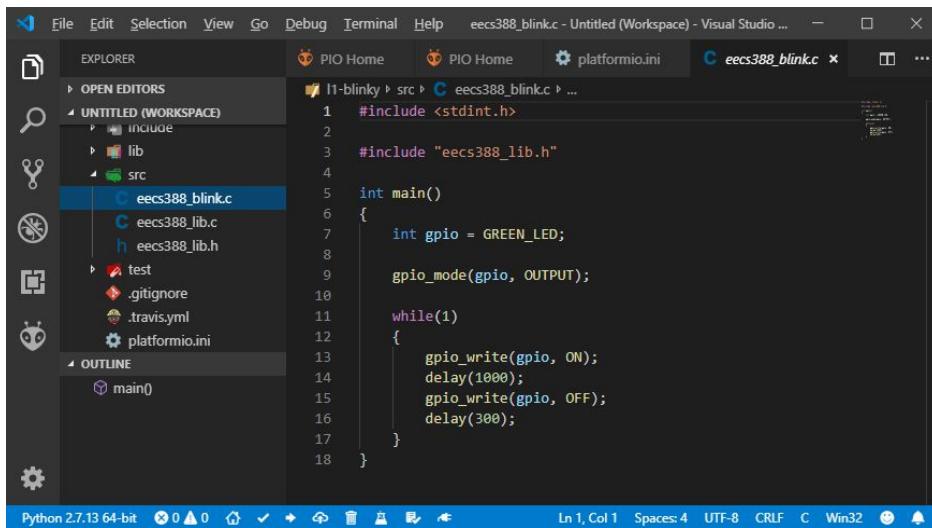
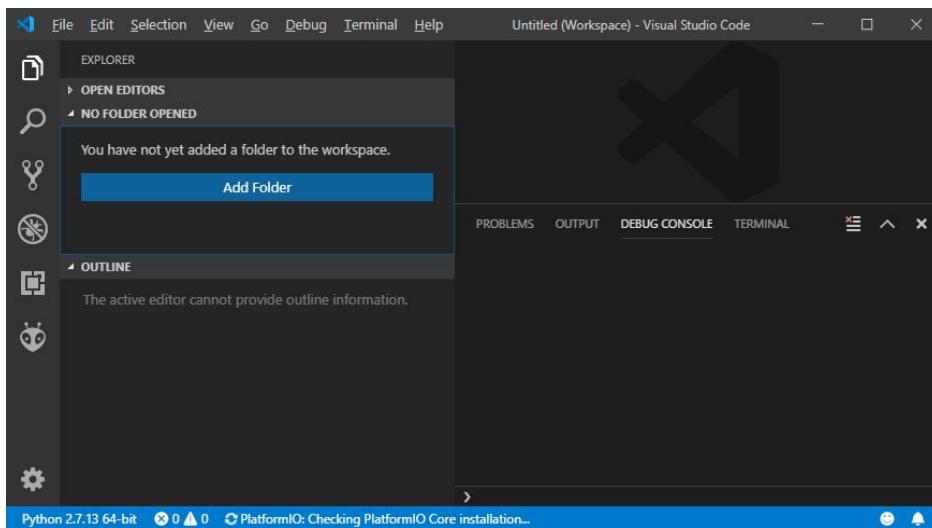
### Task 2.1. Setup a project.

Download the sample project as follows.

```
$ mkdir -p ~/Documents/PlatformIO
$ cd ~/Documents/PlatformIO
$ wget https://ittc.ku.edu/~heechul/courses/eecs388/l1-blinky.tar.gz
$ tar zxvf l1-blinky.tar.gz
```

## Task 2.2. Add the project folder

Add the l1-blinky folder in VSCode.



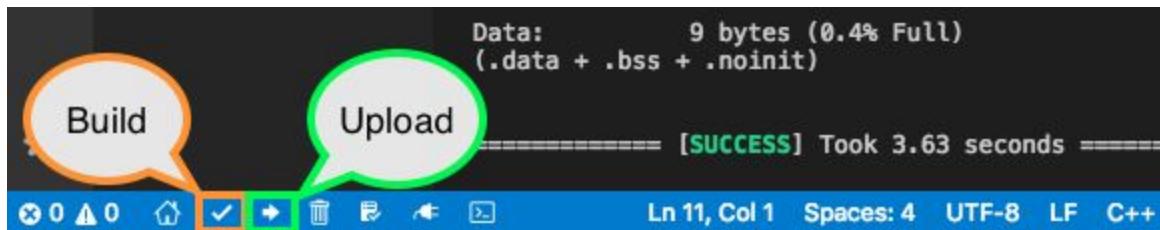
You should be able to see the screen above.

## Task 2.3. Build and Deploy.

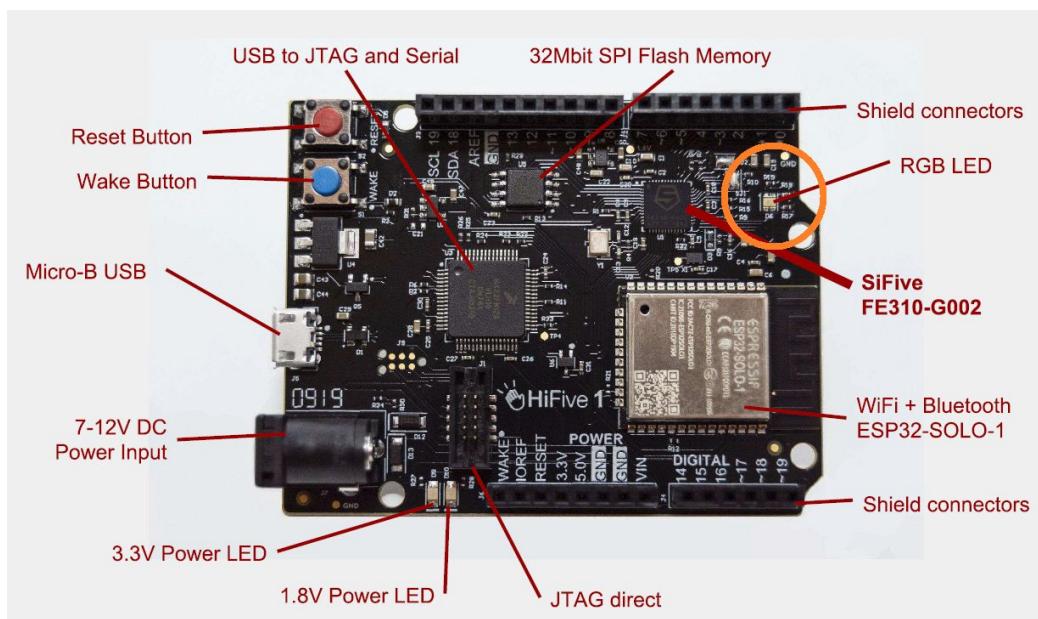
You are now ready to build the code and deploy it on the target board.

First build the program by clicking the build button in the toolbar as shown below or with **ctrl+alt+b** hotkey.

If it is successful, you can now upload the compiled program binary to the board by clicking the upload button or with **ctrl+alt+u** hotkey.



If it was successful, you should see the green led on the board is blinking. (orange circle in the figure below)



## Task 2.4. Debugging (Optional)

The HiFive1 board is already equipped with a hardware debugger support. Thus, you can utilize PlatformIO+VSCode's debugging capability to debug your code. To use debugging, find the 'Debug' menu from the pull down menu or hit 'F5' (or 'Ctrl+F5'). You should be able to see something like the following screen. Then, you can use the debugger toolbar (right top corner of the screenshot) to navigate the code.

eeecs388\_blink.c - Untitled (Workspace) - Visual Studio Code

File Edit Selection View Go Debug Terminal Help

DEBUG PIO Debug (l1-blinky) ▾

VARIABLES

- Local
- Global
- Static

CALL STACK PAUSED ON BREAKPOINT

main@0x20010182 src/eeecs388\_blink.c 9

PERIPHERALS

- AON [0x10000070]
- BACKUP [0x10000080]
- CLINT [0x02000000]
- GPIO0 [0x10012000]
  - VALUE [0x0] = 0x00000000
- WATCH
- BREAKPOINTS
- DISASSEMBLY
  - Disassemble function
  - Switch to assembly

C eeecs388\_tfmini.c eeecs388\_blink.c x main.dbgasm

l1-blinky ▾ src ▾ eeecs388\_blink.c ▾ ...

```
1 #include <stdint.h>
2
3 #include "eeecs388_lib.h"
4
5 int main()
6 {
7     int gpio = GREEN_LED;
8
9     gpio_mode(gpio, OUTPUT);
10
11     while(1)
12     {
13         gpio_write(gpio, ON);
14         delay(1000);
15         gpio_write(gpio, OFF);
16         delay(300);
17     }
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Writing register (pc = 0x20010000)
Transfer rate: 1157 KB/sec, 948 bytes/write.
PlatformIO: Initialization completed
PlatformIO: Resume the execution to `debug_init_break = tbreak main`
PlatformIO: More configuration options -> http://bit.ly/pio-debug
Starting target CPU...
Reading all registers

Temporary breakpoint 1, main () at src/eeecs388_blink.c:9
9         gpio_mode(gpio, OUTPUT);
```

master\* 0 0 0 PIO Debug (l1-blinky) ▾ main() Ln 9, Col 1 Spaces: 4 UTF-8 LF C Linux

## Appendix

# PlatformIO Keybindings

- **ctrl+alt+b / cmd-shift-b / ctrl-shift-b** Build Project
  - **cmd-shift-d / ctrl-shift-d** Debug project
  - **ctrl+alt+u** Upload Firmware
  - **ctrl+alt+s** Open [Serial Port Monitor](#)

## PlatformIO Toolbar



1. [PlatformIO Home](#)
  2. PlatformIO: Build

3. PlatformIO: Upload
4. [PIO Remote](#)
5. PlatformIO: Clean
6. [PIO Unit Testing](#)
7. Run a task... (See “Task Runner” below)
8. [Serial Port Monitor](#)
9. PIO Terminal

PlatformIO documentation

<https://docs.platformio.org/en/latest/ide/vscode.html#installation>