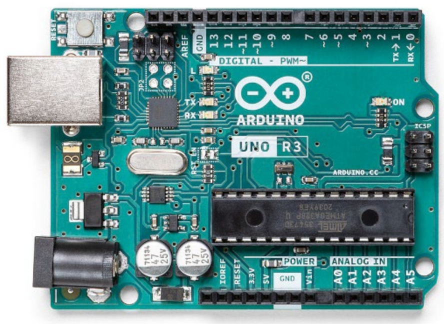
EECS 388 Lab #2

Introduction to Hardware and Integrated Development Environment (IDE)

During the semester, you will develop on three different embedded platforms. The Arduino Uno and HiFive1 Microcontroller are considered bare-metal systems, without an operating system. The Raspberry PI 4 is the third system is considered a SBC (single board computer). During the semester, you will interface the embedded platforms with different sensors, actuators, and LEDs to develop components of a small self-driving car system. You will learn the fundamental concepts and practical skills to design and implement an embedded system.

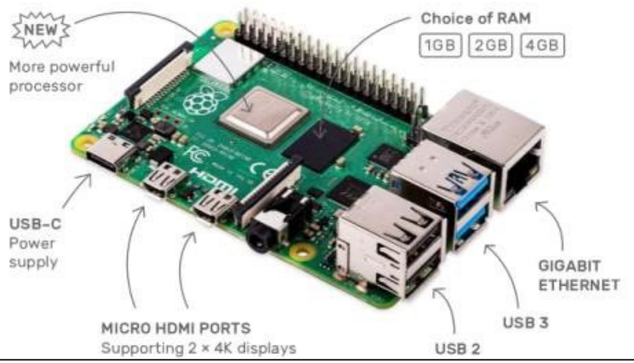
Hardware Platforms

We will use two bare metal embedded platforms and one single-board computer (SBC), shown below.



(a) **Arduino Uno Rev3 –** Used for today's lab

(b) HiFive1 Microcontroller



(c) Raspberry Pi 4

The first platform is an Arduino UNO which uses an 8-bit AVR Instruction Set Architecture (ISA). It uses the 16MHz ATmega328P microcontroller.

The second is an Arduino-compatible microcontroller platform featuring a RISC-V ISA called HiFive1, which will be responsible for basic control and safety of the car,

The third is a Raspberry Pi 4, which will be responsible for angle-based steering using deep learning. In the first half of the semester, we will use Arduino and HiFive1, while in the second half of the semester, we will use the HiFive1 and Raspberry Pi 4 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.

Arduino Uno R3: https://store-usa.arduino.cc/products/arduino-uno-rev3
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/

Task 1: Setup development environment

For software development on the microcontroller, we will use a combination of Visual Studio Code (VSCode) and the PlatformIO IDE. 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

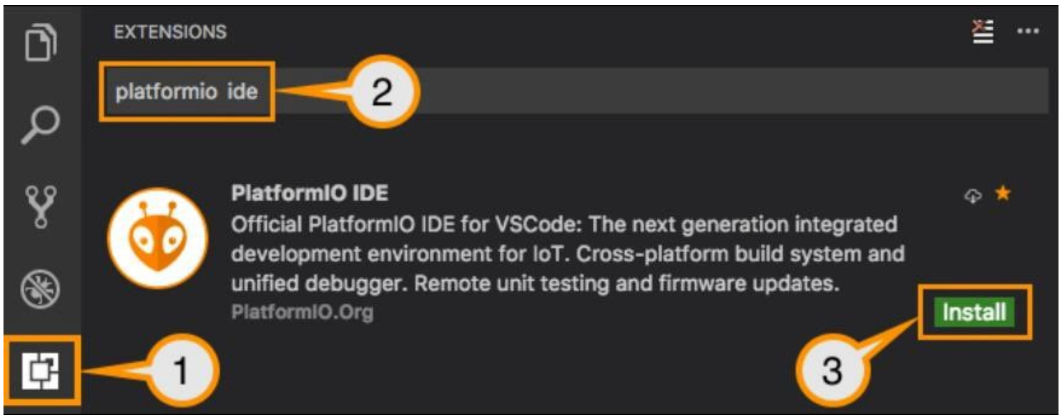
Launch Visual Studio Code from the command line.

```
$ code
```

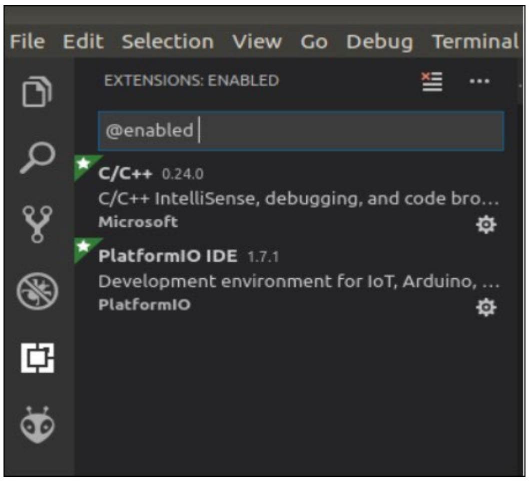
Task 1.2: Install PlatformIO extension

Next, install the PlatformIO IDE extension for VSCode.

1. Open VSCode Extension Manager (keyboard shortcut is ctrl+shift+x)
2. Search for the “platformio ide” extension
3. Install PlatformIO IDE

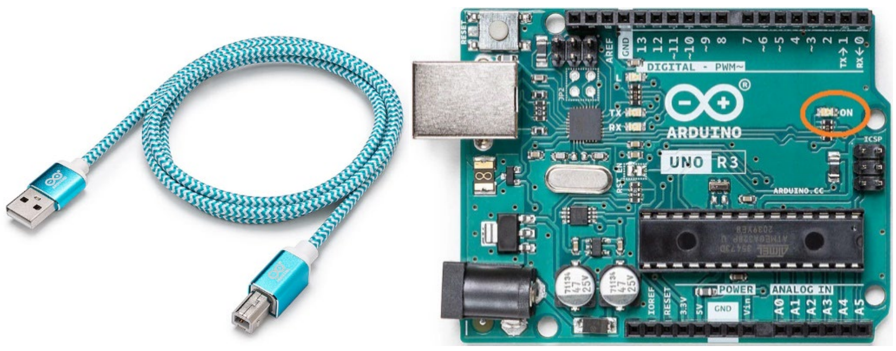

EXTENSIONS
platformio ide
2
PlatformIO IDE
Official PlatformIO IDE for VSCode: The next generation integrated development environment for IoT. Cross-platform build system and unified debugger. Remote unit testing and firmware updates.
PlatformIO.Org
Install
1
3

Install PlatformIO IDE. After the installation is completed, check if you have both 'PlatformIO IDE' and 'C/C++' extensions installed by typing "@enabled" in the search bar. Optionally, installing 'vscode-icons' and 'vscode-pdf' extensions is also recommended.


File Edit Selection View Go Debug Terminal
EXTENSIONS: ENABLED
@enabled
C/C++ 0.24.0
C/C++ IntelliSense, debugging, and code bro...
Microsoft
PlatformIO IDE 1.7.1
Development environment for IoT, Arduino, ...
PlatformIO

Task 1.3: Connect the board to your PC

Next, connect your board to one of your PC's USB ports with the USB2.0 Type A to B cable. Once the board is connected to the PC, one ON yellow power LEDs should be turned on. (See the figure below).



Task 1.4: Run your first program on the Arduino board

Setup a project and download the files by doing the following:

1) `$ mkdir -p ~/Documents/PlatformIO`
2) `$ cd ~/Documents/PlatformIO`

3) Download the zip file from Canvas (**lab02.zip)**

a. Log into Canvas and go to the EECS 388 Embedded Systems course.

b. Inside the "Files>Labs>Lab 2" folder, go to "Lab 2".

c. Click on the **lab02.zip** file to download the source code for lab 2.

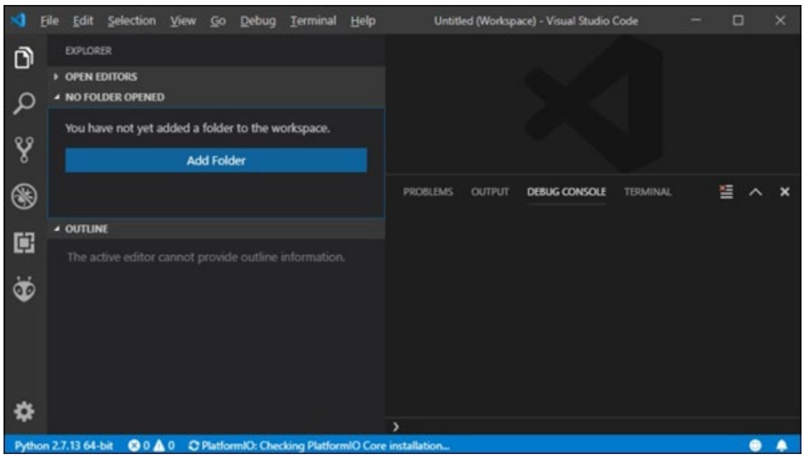
d. Extract the file inside a folder on your PC. You can extract it by:

   i. using the File Explorer and right clicking it **OR**

   ii. typing this command into the terminal (use -d target_directory to extract to a specific directory):

```
$ unzip lab02.zip
```

4) Add the lab02 folder in VSCode. You should be able to see it on the screen below.


File Edit Selection View Go Debug Terminal Help
Untitled (Workspace) - Visual Studio Code
EXPLORER
OPEN EDITORS
NO FOLDER OPENED
Add Folder
OUTLINE
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Python 2.7.13 64-bit

You have not yet added a folder to the workspace.

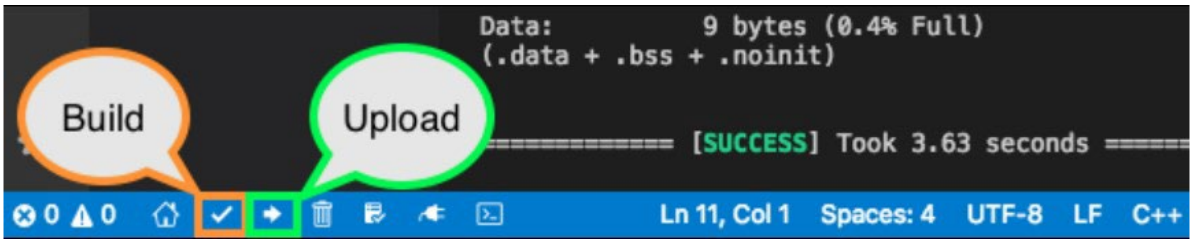
The active editor cannot provide outline information.

```
PlatformIO: Checking PlatformIO Core installation...
```

5) 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 shortcut.

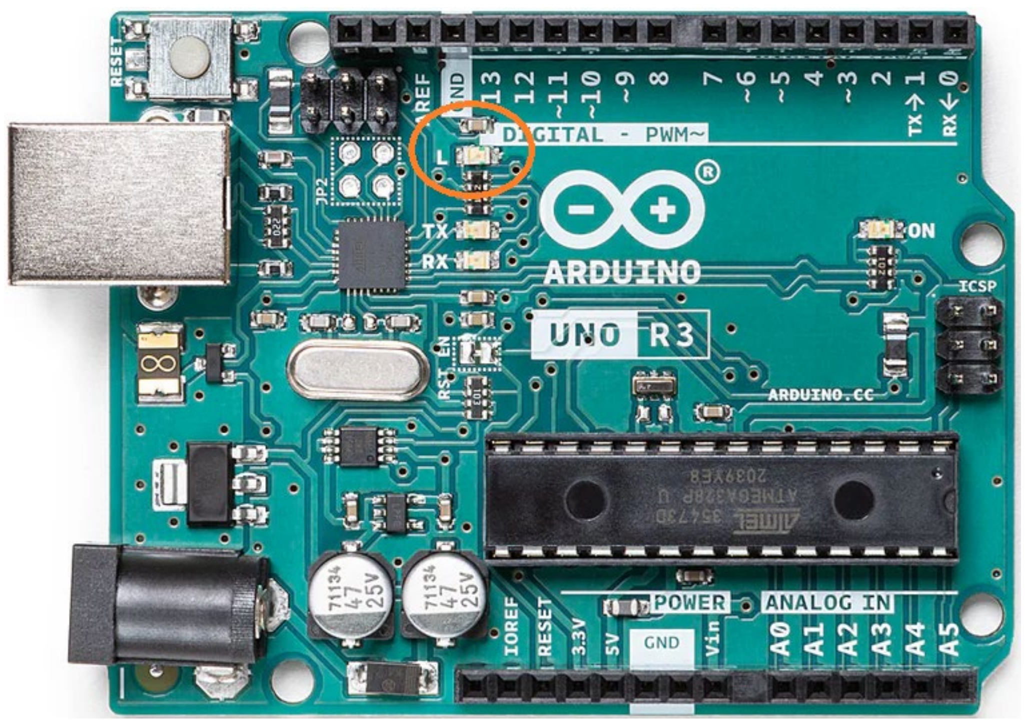
If it is successful, you can now upload the compiled program binary to the board by clicking the **upload** button or with ctrl+alt+u shortcut.



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

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

If it successfully uploaded, you should see the yellow led on the board blinking. (orange circle in the figure below)



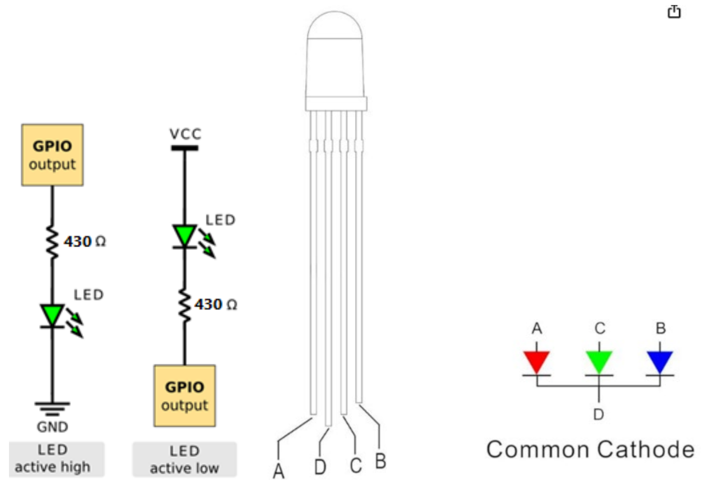
Task 2: RGB Blinky

In this task, you will build an active-high LED circuit on a breadboard and extend the lab02 starter code from Task 1 to make an LED blink red, green, and blue instead of just using the internal Arduino LED.

Follow these instructions to complete **Task 2**.

<u>Task 2.0: Setup active-high LED circuit</u>

Using your breadboard, setup an active-high circuit using the digital GPIO_13 (Red-A), GPIO_12 (Green-C), and GPIO_11 (Blue-B) lines from the Arduino. Use the 430 ohm resistors. Have your GTA inspect your breadboard before you load any code.



Task 2.1: Review the EECS388 library

Review the EECS388 library header (eecs388_lib.h) and the implementation (eecs388_lib.c) to know which library functions are available for your implementation.

Task 2.2: Implement RGB blinky

Modify the code (eecs388_blink.c) according to the code comments to make the RGB LEDs blink. Red LED on, delay 500ms, Red LED off, delay 300ms, Green LED on, delay 500 ms, Green LED off, delay 300ms, Blue LED on, delay 500ms, Blue LED off, delay 300ms…Repeat.

Task 2.3: Add an extra color (OPTIONAL Bonus) - 1 additional point

Modify your code so that the LED also blinks white in the loop, so the color order will be:
red -> blue -> green -> white -> back to red

Task 2.4: Create an active-low LED circuit (OPTIONAL Bonus) - 1 additional point

Add an additional LED and resistor to your breadboard and use GPIO_10 to create an active-low LED in the sequence. Make this light up with a red LED light.

Submission - 40 points (Max 43/40 with bonus)

Milestone: To obtain the 10% (4 points) milestone marks for this lab, you must have Visual Studio Code setup with platformIO installed. You must be able to have the board properly connected and be able to load the first program to the Arduino board. You do not need to have a fully functional program to blink LEDs to get these points.

Demo: To obtain the 40% (16 points) marks for this lab, you must show your TA the working LED program with the LEDs blinking in the correct order. Your TA will also ask you questions about your understanding of the code.

Possible questions:

- Is the GPIO configured as an input or output?
- What is the logic level for turning on the LED?
- Is the code using active high or active low?
- How much current is passing through the LEDs when they are turned on? Assume there is no forward voltage lost across the LED when turned on. Assume the GPIOs provide 5V when outputting.
- What size resistor would you use if you wanted 15 milliamps of current for a brighter LED? Assume there is no forward voltage lost across the LED when turned on.

Code: To obtain the 50% (20 points) go to the "Assignments" section on Canvas and submit your modified **eecs388_blink.cpp** file to the Lab02 submission link for your specific lab section.

Bonus:

To obtain the 2.5% (1 points) you need to submit the files to Canvas by the end of your lab period. You must demo your working code on the board to your GTA before submitting the code to canvas.

You can also obtain an additional 2.5% (1 point) by implementing the extra white color in task 2.3 and demo it to your GTA. (Task 2.3)

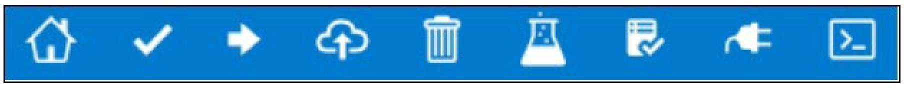
You can also obtain an additional 2.5% (1 point) by creating the active-low LED circuit and using GPIO_10 to light up an additional red LED. (Task 2.4)

Appendix

PlatformIO Key bindings

- 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/integration/ide/vscode.html#installation