EE4144 – Intro to Embedded Systems

Lab Exercise 1

Introduction to PlatformIO IDE

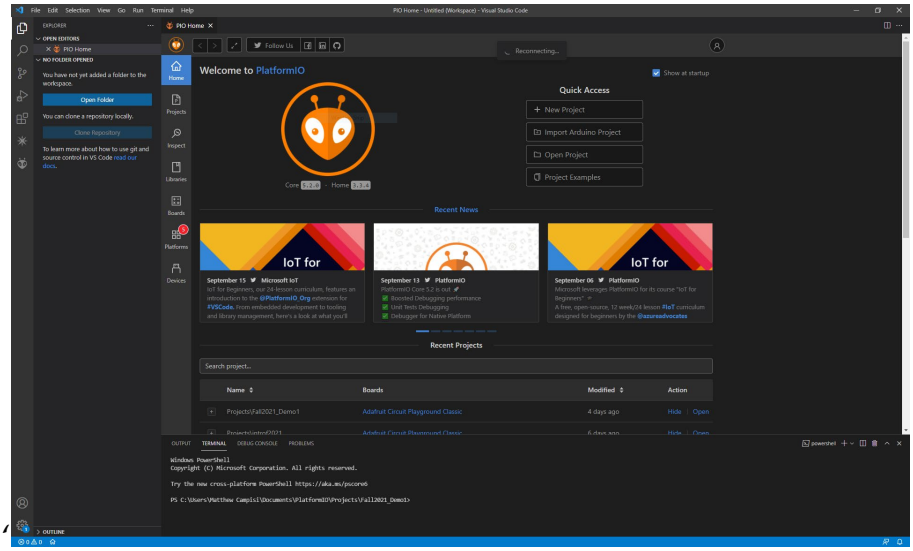
Debugging the Blink Program

Objective:

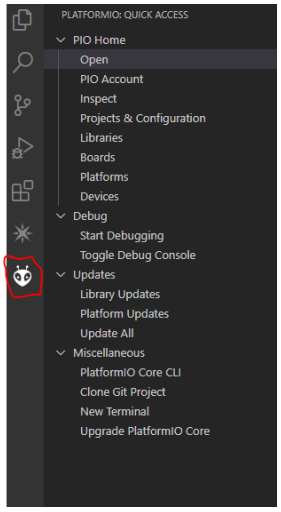
This exercise focuses on implementing a simple embedded program on the Atmel328P/32U4 microcontroller using the Arduino framework and ANSI C. The primary objective is to demonstrate the differences in programming languages and understanding the advantages and disadvantages of each approach. The exercise concludes with examples of how to use the IDE to help debug source code.

Setup:

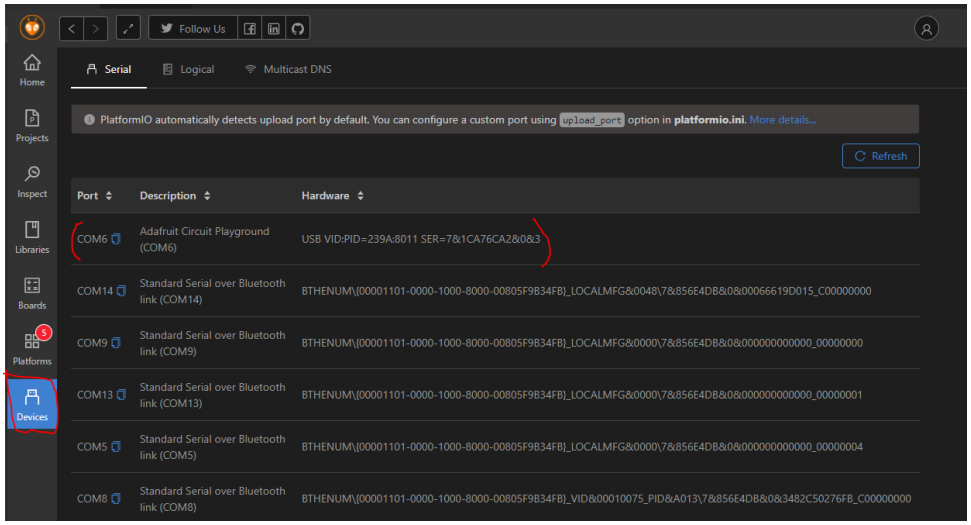
1. Download the both latest PlatformIO IDE and VS Code from: https://platformio.org/install/ide?install=vscode
   This is the development environment used to create, write, compile, and upload the embedded software.
2. Confirm that you can run the VS Code app and the PlatformIO add on loads as a tab in VS Code. See below.



3. If the PlatformIO tab does not load automatically, click on the PlatformIO link->PIO Home->Open.

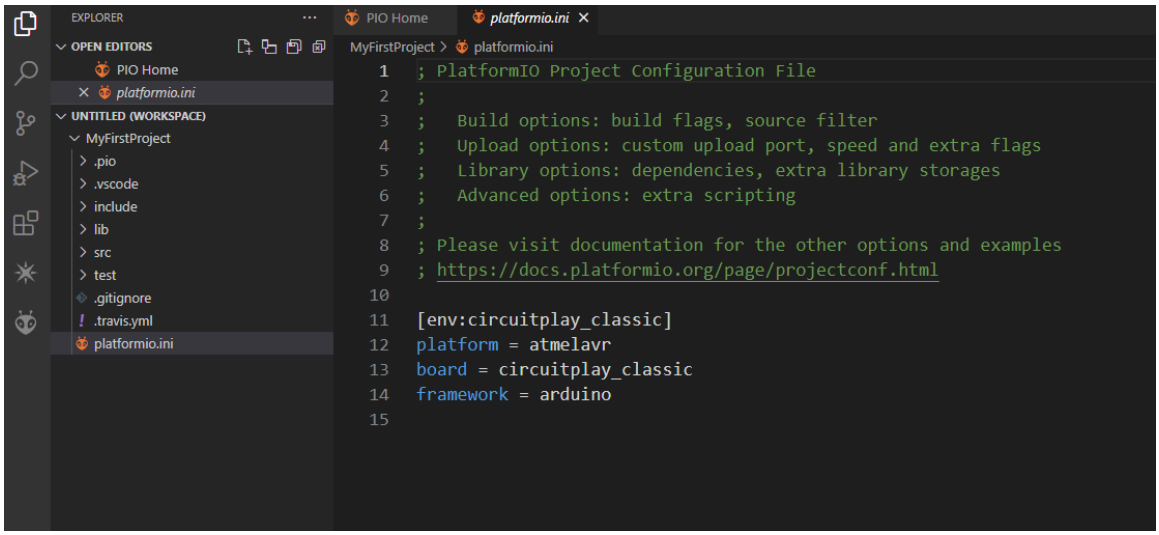


4. Plug in your Adafruit Playground Classic board using a micro USB data cable. The green LED located near the USB port should illuminate. The default demo app should cycle through the LEDs indicating it is a new board.
5. Confirm your device is recognized by pressing "Devices" and confirm there is an entry for the Adafruit Playground Classic. Your COMX will indicate the port number, in this case, COM6.



Running the Blink Sketch:

1. Click back on "Home" in the PlatformIO tab of VS Code. Then click New Project to launch the project wizard.
2. For project Name enter "MyFirstProject"
3. Click on the board field and type "classic". The Adafruit Playground Classic should appear in the list. Select that board.
4. Leave the Framework as Arduino and press "Finish". The following workspace will appear.



5. Now, click on "main.cpp" under UNTITLED (WORKSPACE)->src.
6. The following blank sketch will appear:

```
#include <Arduino.h>

void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

7. Replace the text line 2 to line 9 with the following code. Be sure to leave line 1 as is.

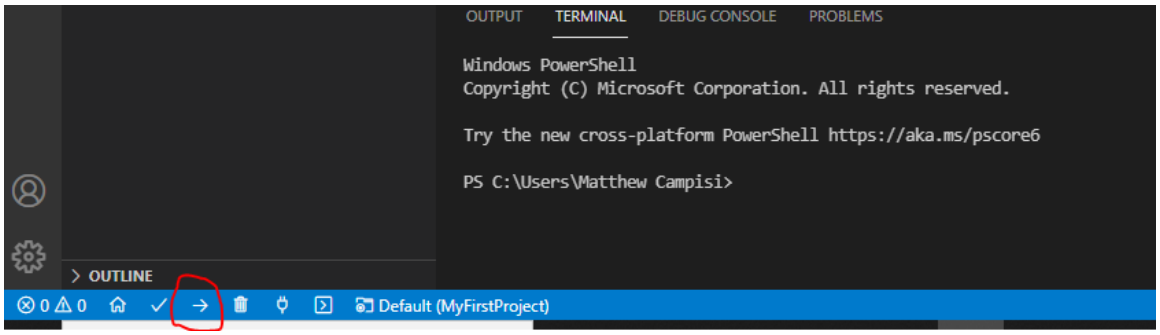
```
int ledPin = 13;
void setup ()
{
      pinMode(ledPin, OUTPUT);
}

void loop ()
{
      digitalWrite(ledPin, HIGH);
      delay(1000);
      digitalWrite(ledPin, LOW);
      delay(1000);
}
```

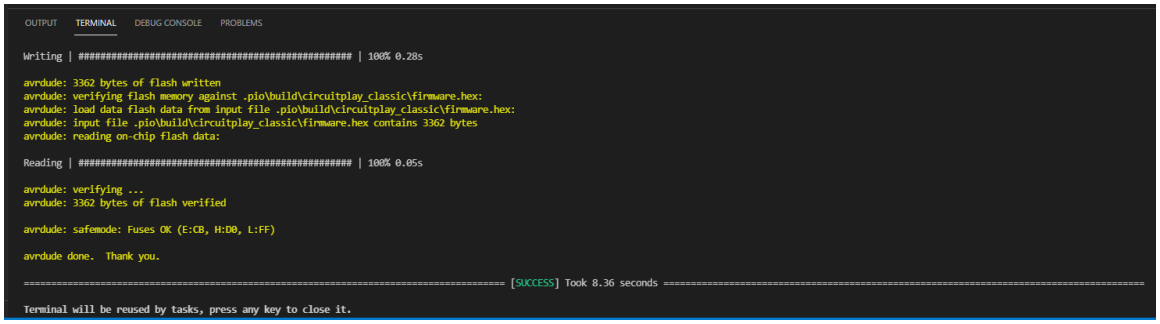
Regarding this example, all programs built using the Arduino framework include the two functions **setup()** and **loop().** As the labels are meant to imply, the **setup()** function is run once when the program begins, so initialization would take place here, while the **loop()** function executes as an infinite loop until the processor is reset. This example includes a global variable **ledPin**. When the program is compiled on the host computer, this particular location in the target memory is initialized with the value 13 because the Adafruit Classic circuit board's 13th I/O pin has the red LED labeled "#13" on the board (next to the USB port). When **setup()** is run at the start of the program, a single library call is made to make the 13th I/O pin, referenced via that named variable, as an output pin, so that the processor can drive the LED either high or low, hence turning the LED on or off. Within the **loop()** function, the LED is turned on, and then a fixed delay of 1000 milliseconds is executed before the LED is turned off, and another delay.

Notice that none of the functions presented in this example are either assembly nor are they ANSI C; instead, they are all part of the Arduino library of functions.

8. With the source code in main.cpp, press the download icon located on the lower tool bar icon in order to both compile the program and download the program into the board.



9. A number of files will be compiled and you will notice files scrolling through the terminal window. This is normal. Upon a successful download, the terminal window will indicate "Success" (in green).



→(1)Demonstrate the successful upload to the instructor.

10. Now that you have compiled and uploaded your first program, let's convert it from the easy-to-use-yet-non-standard library calls to easy-once-you-know-them-and-portable ANSI C instructions.
Change the source code to the following: (again leave line 1 as is)

```
void MyDelay (unsigned long  mSecondsApx);

void setup ()
{
      unsigned char *portDDRC;
      portDDRC = (unsigned char *) 0x27;
      *portDDRC |= 0x80;

}

void loop ()
{

      unsigned char *portC;
      portC = (unsigned char *) 0x28;
      *portC |= 0x80;
      MyDelay(1000);
      *portC &= 0x7F;
      MyDelay (1000);
}

void MyDelay (unsigned long mSecondsApx)
{
      volatile unsigned long i;
      unsigned long endTime = 250 * mSecondsApx;

      for (i = 0; i < endTime; i++);
}
```

→(2)Demonstrate the successful upload of the new program
→(3)Explain what each line of setup() and loop() does.
→(4)Why is "|=" used in one line of loop(), but "&=" is used in the other?
→(5)Explain exactly what MyDelay() does.

Further Experiments:

1. Modify the provided example Blink sketch to cause the LED to blink in the following pattern:
    a. Blink once
    b. Pause for one second

c. Blink twice
d. Pause for one second
e. Blink 3 times
f. Pause for one second and then start over.

→(6)Show the instructor your code.

2. What is the total number of bytes required by your program above. Obtained this value after you compile your program in the bottom window of the editor.
3. Now modify the ANSI C version of the Blink program to cause the LED to blink in the pattern described in 1.

→ (7)Show the instructor your code.

4. What is the total number of bytes required by your program above (in 3).
5. What can you conclude about the required bytes for each program?

Debugging:

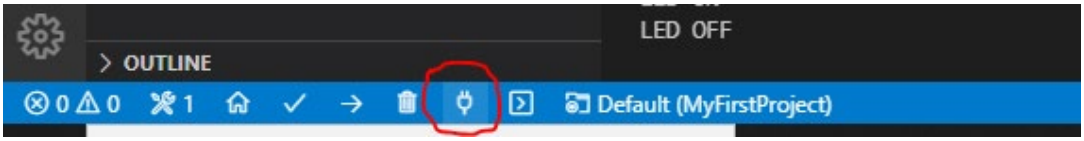
Run-time debugging often utilizes the **Serial.print()** and **Serial.println()** functions in Arduino. Modify the Blink Arduino sketch to include the following in the **setup()** function:

```
Serial.begin(9600);
```

Now modify the **loop()** function to the following:

```
void loop()
{
      digitalWrite(13, HIGH);
      Serial.println("LED ON");
      delay(1000);
      digitalWrite(13, LOW);
      Serial.println("LED OFF");
      delay(1000);
}
```

1. Download the modified code to the board. After a successful download, confirm that the LED" is blinking. Now, open the serial monitor in the IDE by pressing the serial terminal icon on the lower toolbar of the IDE.



2. Now observe the output in the serial window. It should be displaying "LED ON" and "LED OFF" respectively.

→(8)Show the instructor the serial output window.

3. Practice *compile-time* debugging by fixing all of the syntax errors in the following code. Use the IDE compile output information to help identify the errors. Note the required changes for the program to compile successfully.

```
void MyDelay (unsigned long mSecondsApx);

void setup ()
{

      unsigned char  *portDDRC;

      portDDRC = (unsigned char *) 0x27;

      *portDDRC |= 0x80;
}

void  loop ()
{

      unsigned char  *portC;

      portC = (unsigned char *) 0x28;

      *portC |= 0x80;
      MyDelay{1000}
      portB &= 0x7F:
      MyDelay [1000],
}
```

```
void MyDelay (unsigned long  mSecondsApx)
{
      volatile unsigned long  i;
      unsigned long endTime = 1000 * mSecondsApx;

      for (i = 0; i < endTime; i++);
}
```

→(9)Show the instructor the successful compile

4. Below are three ANSI C programs (A., B., and C.). Practice run-time debugging by fixing each so that the LED will blink correctly (once per second). Be sure to leave Line 1 as is!

Program A.

```
      void NewDelay (unsigned char mSecondsApx);

      void  setup()
      {
            unsigned char *portDDRC;

            portDDRC = (unsigned char *) 0x27;

            *portDDRC |= 0x80;
      }

      void loop ()
      {
            unsigned char  *portC;
            portC = (unsigned char  *) 0x28;
            *portC |= 0x20;
            NewDelay (100);
            *portC %= 0x7F;
            NewDelay (100);
      }
      void NewDelay (unsigned char mSecondsApx)
      {
            volatile unsigned char i;
            unsigned long endTime = 1000 * mSecondsApx;
            for  (i = 0; i < endTime; i++);
      }
```

Program B.

```
void NewDelay (unsigned long mSecondsApx);
void setup()
{
      unsigned char *portDDRC;
      portDDRC = (unsigned char *) 0x27;
      *portDDRC |= 0x80;
}
void loop ()
{
      unsigned char *portC;
      portC = (unsigned char *) 0x28;
      *portC |= 0x80;
      NewDelay (100);
      *portC &= 0xDF;
      NewDelay (100);
}
void NewDelay (unsigned long mSecondsApx)
{
      volatile unsigned long i;
      unsigned char j;
      unsigned long k;
      unsigned long endTime = 100 * mSecondsApx;

      for (i = 0; i < endTime; i++)
      {
            j = 10;
            do
            {
                  j = j  -  i;
                  k = i/j;
            } while (k>0);
      }
}
```

Program C.

```
void  NewDelay (unsigned long mSecondsApx);
void  setup()
{
      unsigned char *portDDRC;

      portDDRC = (unsigned char *) 0x24;

      *portDDRC |= 0x20;
}
void loop ()
{
      unsigned char *portC;

      portC = (unsigned char *) 0x25;

      *portC |= 0x20;
      NewDelay (100);
      *portC &= 0x7F;
      NewDelay (100);
}
void NewDelay (unsigned long mSecondsApx)
{
      volatile unsigned long i;
      unsigned char  j = 0;
      unsigned long endTime = 100 * mSecondsApx;
      i= 0;
      while (j = 0)
      {
            i++;
            if (i = endTime)
            {
                  j = 1;
            }
      }
}
```

→(10) Explain to the instructor the changes made to Program A.
→(11) Explain to the instructor the changes made to Program B.
→(12) Explain to the instructor the changes made to Program C.