Engineering Integration I

Student : Ali Behbehani

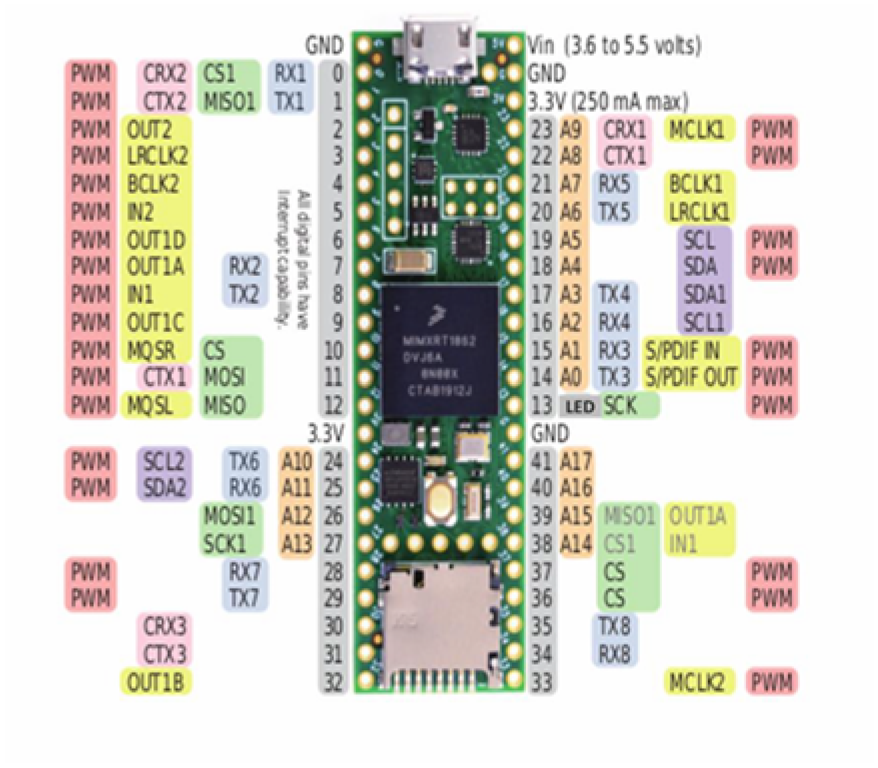
Course : Engineering Integration I

Section : Friday 1:00 - 3:50

Introduction

This laboratory session provided an introduction to the fundamental operations and programming of the Teensy 4.1 microcontroller. The lab was structured around five experiments, each intended to cultivate fundamental competencies in embedded systems. The initial four experiments concentrated on core proficiencies, including circuit construction on a breadboard, programming digital and analog input/output devices, and sensor utilization. The concluding experiment necessitated the integration of multiple concepts to develop a functional embedded system.

A critical skill in microcontroller applications is the ability to interpret pin diagrams and establish correct circuit connections. The Teensy 4.1 pin diagram offers details regarding power pins, ground connections, and programmable pins, which can be configured for either input or output functions. These proficiencies are crucial for subsequent projects involving automation, sensing systems, and electronic device control.



Pre-Lab

Resistor Color Codes

330Ω, 4-band, ±5%
Orange – Orange – Brown – Gold

10kΩ, 5-band, ±1%
Brown – Black – Black – Red – Brown

Microprocessor vs Microcomputer vs Microcontroller

A microprocessor is a processor chip that requires external memory and input/output hardware to operate. A microcomputer contains a processor along with memory and I/O components in a complete system. A microcontroller integrates a processor, memory, and input/output peripherals onto a single chip.

Microcontrollers are ideal for embedded applications because they are compact, efficient, and designed to directly interface with hardware devices such as sensors, LEDs, and switches.

Experiment 1 - Teensy LED Blink

Objective

The objective of this experiment was to validate the proper connection of the Teensy microcontroller to the computer and its capacity to execute uploaded programs.

Procedure

The Teensy board was interfaced with the computer via a USB cable. Subsequently, the Arduino IDE was initiated, and the Teensy 4.1 board was selected. The BlinkWithoutDelay example program was accessed and modified to utilize the onboard LED pin. The program was then compiled and uploaded to the board.

Results

The onboard LED exhibited blinking behavior at one-second intervals, thereby confirming successful communication between the Teensy and the computer.

Conclusion

This experiment confirmed the correct functionality of the Teensy board, indicating its readiness for subsequent programming and circuit experiments.

Experiment 2 - RGB LED Interface

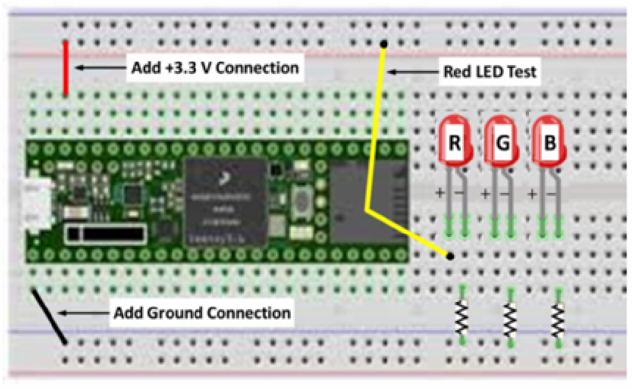
Experiment 2(a) - Digital RGB LED Blink

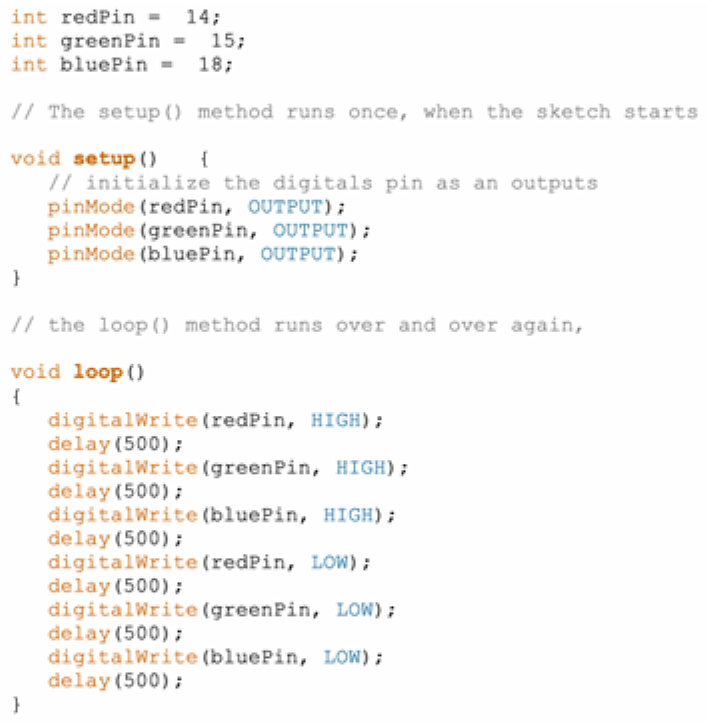
Objective

The goal of this experiment was to understand how to connect and control several LEDs using digital output signals from the microcontroller.

Procedure

Red, green, and blue LEDs were connected to the breadboard, with 330Ω resistors used to limit current. The LEDs were then connected to PWM-capable pins on the Teensy microcontroller. A digital output program was created and uploaded to the Teensy board, which turned each LED on and off in a specific order.





```
int redPin =  14;
int greenPin =  15;
int bluePin =  18;

// The setup() method runs once, when the sketch starts

void setup()   {
   // initialize the digitals pin as an outputs
   pinMode(redPin, OUTPUT);
   pinMode(greenPin, OUTPUT);
   pinMode(bluePin, OUTPUT);
}

// the loop() method runs over and over again,

void loop()
{
   digitalWrite(redPin, HIGH);
   delay(500);
   digitalWrite(greenPin, HIGH);
   delay(500);
   digitalWrite(bluePin, HIGH);
   delay(500);
   digitalWrite(redPin, LOW);
   delay(500);
   digitalWrite(greenPin, LOW);
   delay(500);
   digitalWrite(bluePin, LOW);
   delay(500);
}
```

Results

Each LED turned on and off as expected, following the programmed sequence.

The sequential blinking of the red, green, and blue LEDs confirmed the proper functioning of the digital output.

Program Overview

The RGB_DigitalOutput program initializes each LED pin as an output within the setup function. The loop function then activates and deactivates each LED in succession, employing digitalWrite commands and delay timing. This process generates a recurring pattern, with each LED illuminating independently.

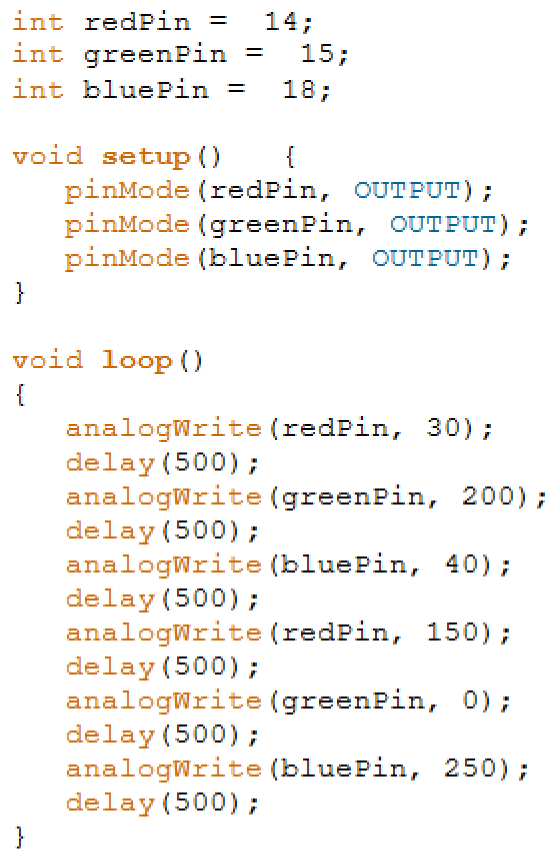
Conclusion

This experiment successfully illustrated the application of digital output signals for LED control, thereby enabling the creation of fundamental lighting sequences through microcontroller programming.

E2.1 RGB DigitalOutput Program Explanation

The RGB_DigitalOutput program governs an RGB LED's operation by cyclically activating and deactivating its red, green, and blue components. Within the setup function, the pinMode function is employed to designate each LED pin as an output. Subsequently, the loop function sequentially turns each LED on via digitalWrite(HIGH), followed by digitalWrite(LOW) to turn it off, with a delay introduced between each action. This process generates a recurring sequence, wherein each LED illuminates independently, thereby illustrating digital output control facilitated by the microcontroller.

Experiment 2(b) - RGB LED Utilizing PWM (Analog Output)



```
int redPin =  14;
int greenPin =  15;
int bluePin =  18;

void setup()   {
   pinMode(redPin, OUTPUT);
   pinMode(greenPin, OUTPUT);
   pinMode(bluePin, OUTPUT);
}

void loop()
{
   analogWrite(redPin, 30);
   delay(500);
   analogWrite(greenPin, 200);
   delay(500);
   analogWrite(bluePin, 40);
   delay(500);
   analogWrite(redPin, 150);
   delay(500);
   analogWrite(greenPin, 0);
   delay(500);
   analogWrite(bluePin, 250);
   delay(500);
}
```

Objective

The aim of this experiment was to investigate the application of PWM signals in modulating LED brightness and generating variations in color intensity.

Procedure

A PWM program was uploaded to the Teensy board. The analogWrite function was employed to transmit PWM signals to each LED pin. Various PWM values were assessed to observe alterations in brightness.

Results

The LED brightness exhibited changes corresponding to the PWM values implemented within the program.

Smooth fading and intensity transitions were observed in the LEDs.

Program Explanation

PWM controls LED brightness by quickly turning the output signal on and off. The brightness depends on the signal's duty cycle. The analogWrite function accepts values from 0 to 255; higher values produce brighter LED output, while lower values produce dimmer output.

Conclusion

This experiment showed how PWM signals allow microcontrollers to control analog-like behavior using digital signals.

E2.2 PWM Brightness Explanation

PWM, is a common method for adjusting LED brightness. It works by quickly turning the LED power on and off. The perceived brightness is dictated by the signal's duty cycle; this is the proportion of time the signal is in the "on" state within each cycle. A longer duty cycle means the LED stays lit for a greater duration, resulting in a brighter light. Conversely, a shorter duty cycle makes the LED dimmer. The rapid switching is key, as it prevents the human eye from detecting any flicker, creating the illusion of a smooth, continuous brightness.

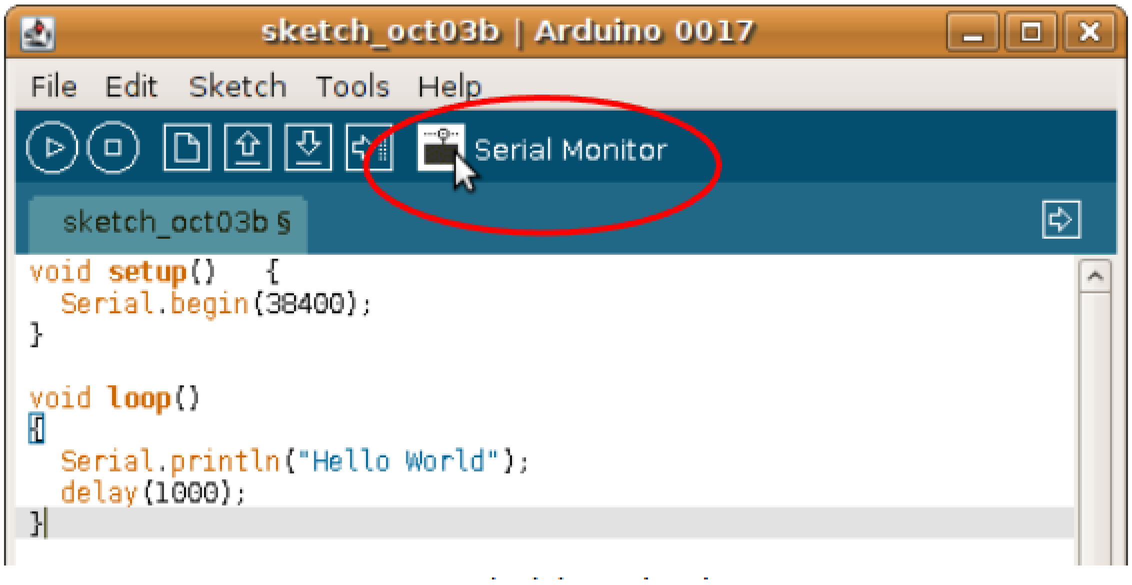
E2.3 PWM Program Explanation

The PWM program uses the analogWrite function to send PWM signals to the LED pins. This function accepts values from 0 to 255. A value of 0 means the LED is off, and 255 means the LED is at its brightest. The program changes these PWM values over time, using delay timing. This allows the LED brightness to change gradually, creating fading and brightness transition effects.

Experiment 3 - Pushbutton and Serial Monitor

Objective

The objective of this experiment was to understand how microcontrollers interpret digital inputs and subsequently transmit data to a computer via serial communication.

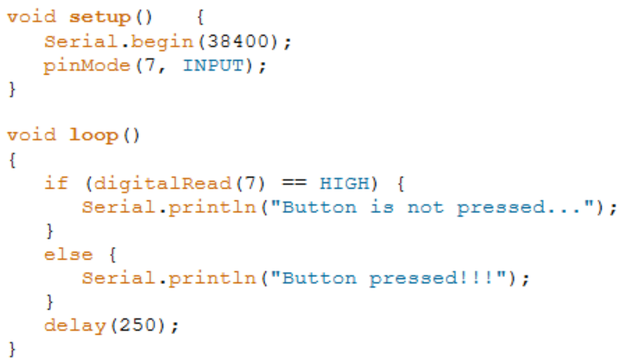


```
void setup()   {
  Serial.begin(38400);
}

void loop()
{
  Serial.println("Hello World");
  delay(1000);
}
```

Procedure

A pushbutton circuit was constructed, incorporating a 10kΩ pull-up resistor. The pushbutton was then connected to a Teensy input pin. A program employing digitalRead was developed to ascertain the button's state. Serial.print statements were utilized to present the button's state on the Serial Monitor.

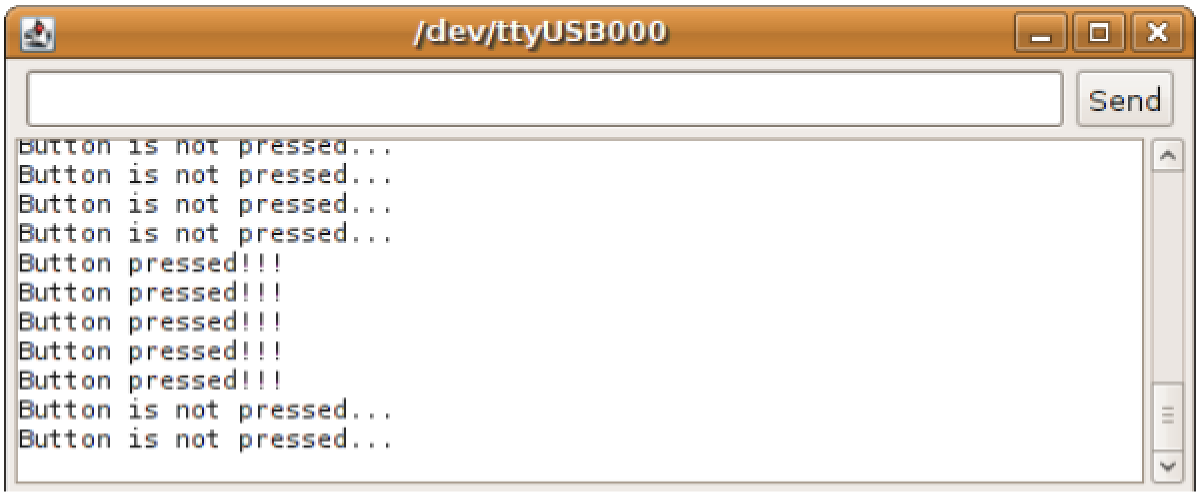


```
void setup()   {
   Serial.begin(38400);
   pinMode(7, INPUT);
}

void loop()
{
   if (digitalRead(7) == HIGH) {
      Serial.println("Button is not pressed...");
   }
   else {
      Serial.println("Button pressed!!!");
   }
   delay(250);
}
```

Results

The Serial Monitor exhibited messages that varied in accordance with the pushbutton's pressed or released condition.



```
Button is not pressed...
Button is not pressed...
Button is not pressed...
Button is not pressed...
Button pressed!!!
Button pressed!!!
Button pressed!!!
Button pressed!!!
Button pressed!!!
Button is not pressed...
Button is not pressed...
```

Conclusion

This experiment demonstrated the application of pushbuttons as digital input devices, and the utility of serial communication in facilitating debugging and monitoring of program functionality.

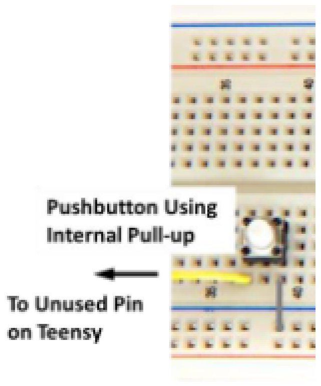
Experiment 4 - Analog Input and Sensors

Experiment 4(a) - Analog Input Using Potentiometer

Objective

The aim of this experiment was to elucidate the process by which a microcontroller acquires and manipulates analog signals, specifically through the utilization of a potentiometer.

```
void setup()    {
   Serial.begin(38400);
   pinMode(8, INPUT_PULLUP);
}
```



Procedure

A potentiometer circuit was assembled and subsequently interfaced with an analog input pin. Voltage readings were then obtained via a program that incorporated the analogRead function.

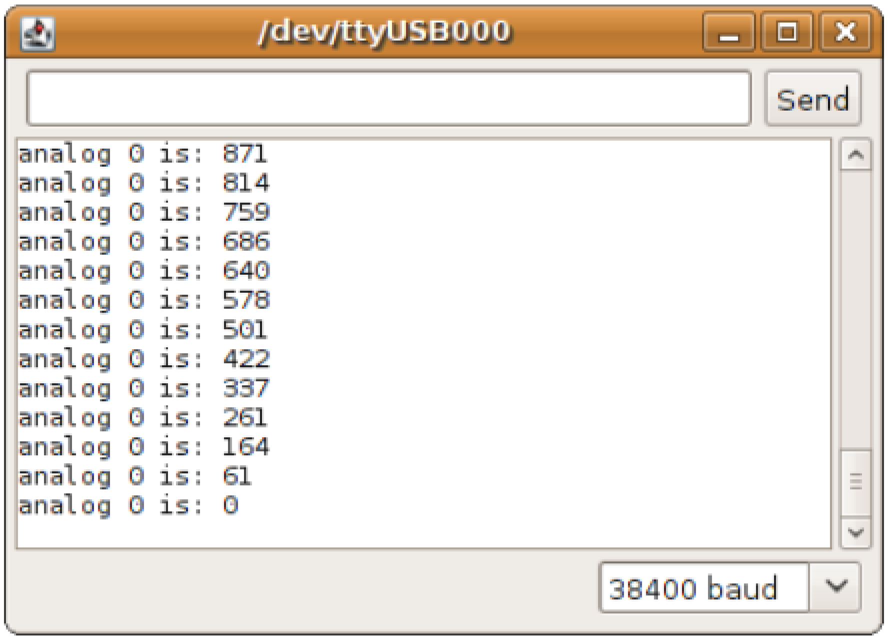
```
void setup()
{
   Serial.begin(38400);
}

int val;

void loop()
{
   val = analogRead(2);
   Serial.print("analog 2 is: ");
   Serial.println(val);
   delay(250);
}
```

Results

The analog values displayed in the Serial Monitor exhibited fluctuations corresponding to modifications applied to the potentiometer.



```
analog 0 is: 871
analog 0 is: 814
analog 0 is: 759
analog 0 is: 686
analog 0 is: 640
analog 0 is: 578
analog 0 is: 501
analog 0 is: 422
analog 0 is: 337
analog 0 is: 261
analog 0 is: 164
analog 0 is: 61
analog 0 is: 0
```

Program Explanation

The analogRead function yields values spanning from 0 to 1023. This value is subsequently divided by 4 to conform to the 0–255 range mandated by analogWrite. The equation 255 – redIntensity generates an inverse brightness effect across the LEDs.

Conclusion

This experiment served to illustrate the capacity of microcontrollers to interpret real-world sensor signals and subsequently transform them into digital data suitable for processing.

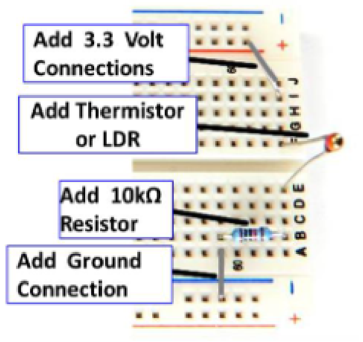
Experiment 4(b) - Analog Input Using Sensor (Thermistor or LDR)

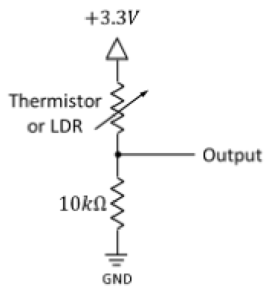
Objective

The aim of this experiment was to demonstrate the generation of analog signals by environmental sensors, which can subsequently be interpreted and manipulated by a microcontroller.

Procedure

A sensor voltage divider circuit, employing either a thermistor or a light-dependent resistor (LDR), was assembled and linked to an analog input pin. The sensor's output values were then observed through a serial output program.





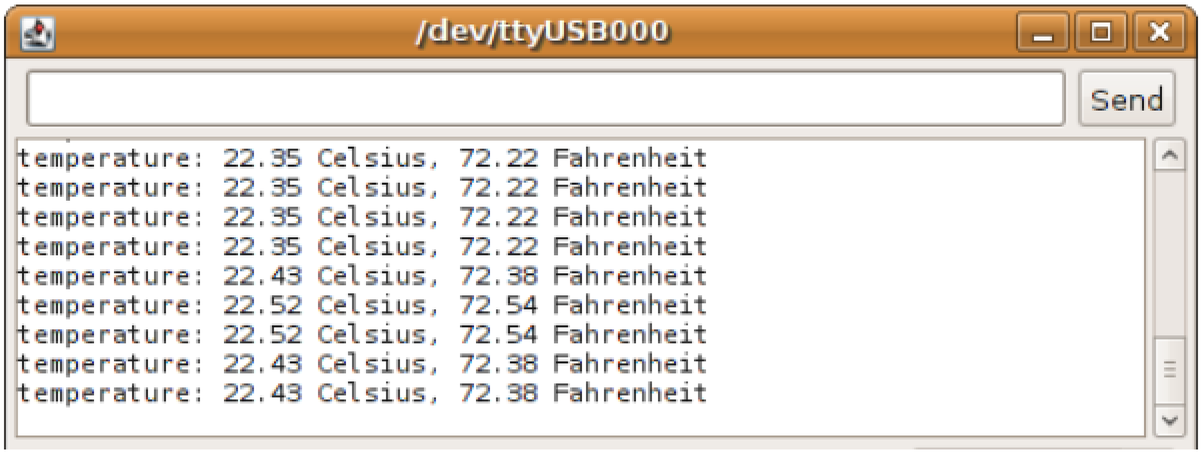
```
void setup()
{
   Serial.begin(38400);
}

float code;
float celsius;
float fahrenheit;

void loop()
{
   code = analogRead(3);
   celsius = 25 + (code - 512) / 11.3;
   fahrenheit = celsius * 1.8 + 32;
   Serial.print("temperature: ");
   Serial.print(celsius);
   Serial.print(" Celsius, ");
   Serial.print(fahrenheit);
   Serial.println(" Fahrenheit");
   delay(1000);
}
```

Results

The output from the sensor circuit displayed variations that corresponded to changes in environmental conditions, including light intensity or temperature.



```
temperature: 22.35 Celsius, 72.22 Fahrenheit
temperature: 22.35 Celsius, 72.22 Fahrenheit
temperature: 22.35 Celsius, 72.22 Fahrenheit
temperature: 22.35 Celsius, 72.22 Fahrenheit
temperature: 22.43 Celsius, 72.38 Fahrenheit
temperature: 22.52 Celsius, 72.54 Fahrenheit
temperature: 22.52 Celsius, 72.54 Fahrenheit
temperature: 22.43 Celsius, 72.38 Fahrenheit
temperature: 22.43 Celsius, 72.38 Fahrenheit
```

Conclusion

This experiment successfully illustrated the integration of sensors within microcontroller systems for the purpose of monitoring environmental conditions and producing real-time data.

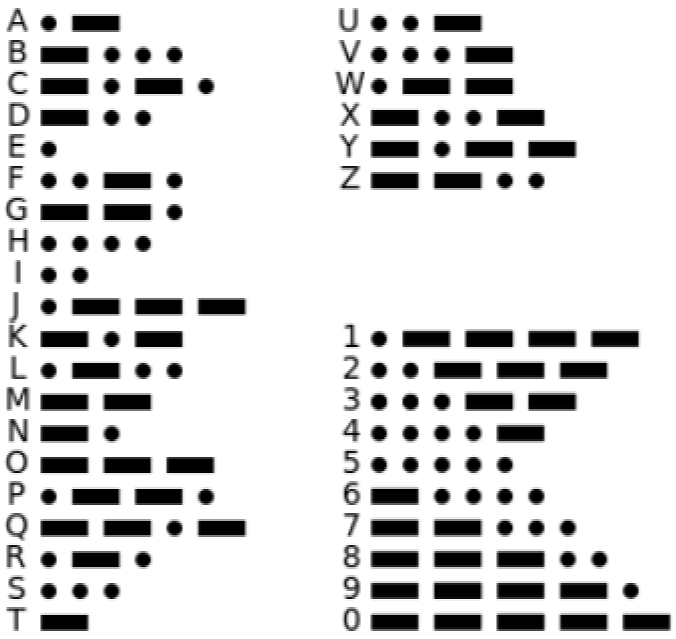
E4.1 - UserControlledLED Program Explanation

The UserControlledLED program utilizes the analogRead function to ascertain the potentiometer's position. This function yields values spanning from 0 to 1023, corresponding to the measured voltage. To accommodate the analogWrite function's value constraints, which are limited to 0-255, this value is scaled down by a factor of four. Consequently, the red LED's brightness is directly modulated by this scaled value. Furthermore, the equation 255 − redIntensity establishes an inverse brightness correlation between the LEDs. Therefore, an increase in the red LED's brightness results in a decrease in the other LED's brightness, thereby producing a fading transition effect.

Experiment 5 - Morse Code SOS System

Objective

The objective of this experiment was to integrate input and output systems, thereby constructing a functional embedded application.



A	● ▬	U	● ● ▬
B	▬ ● ● ●	V	● ● ● ▬
C	▬ ● ▬ ●	W	● ▬ ▬
D	▬ ● ●	X	▬ ● ● ▬
E	●	Y	▬ ● ▬ ▬
F	● ● ▬ ●	Z	▬ ▬ ● ●
G	▬ ▬ ●		
H	● ● ● ●		
I	● ●		
J	● ▬ ▬ ▬		
K	▬ ● ▬	1	● ▬ ▬ ▬ ▬
L	● ▬ ● ●	2	● ● ▬ ▬ ▬
M	▬ ▬	3	● ● ● ▬ ▬
N	▬ ●	4	● ● ● ● ▬
O	▬ ▬ ▬	5	● ● ● ● ●
P	● ▬ ▬ ●	6	▬ ● ● ● ●
Q	▬ ▬ ● ▬	7	▬ ▬ ● ● ●
R	● ▬ ●	8	▬ ▬ ▬ ● ●
S	● ● ●	9	▬ ▬ ▬ ▬ ●
T	▬	0	▬ ▬ ▬ ▬ ▬

Procedure

A pushbutton circuit and an LED circuit were interfaced with the Teensy microcontroller. Subsequently, a program was developed to transmit the SOS distress signal in Morse code upon the activation of the pushbutton.

Results

When I pressed the pushbutton, the LED effectively transmitted the SOS Morse code signal. The timing of the signal adhered to the established Morse code protocol. Furthermore, the system executed the complete SOS sequence prior to resetting, irrespective of the continuous pushbutton activation.

Conclusion

The experiment's outcomes validated the successful integration of digital input and output systems. The microcontroller accurately processed the pushbutton input, subsequently generating a timed LED output. Consequently, this confirms the capability of embedded systems to integrate hardware and software components for the execution of practical signaling and control functions.

Final Conclusion

The laboratory session offered practical exposure to microcontroller programming, circuit assembly, and sensor integration. The conducted experiments illustrated the utilization of digital and analog signals within embedded systems. Furthermore, the lab emphasized the significance of precise wiring, appropriate resistor selection, and accurate signal measurement. These proficiencies are indispensable for practical engineering applications, particularly those pertaining to automation and control systems.

Documentation Statement

I certify that this lab report represents my own work and that all procedures were completed according to the lab instructions and academic integrity guidelines. **(Figures were from lab file)**

Print Name (first last): Ali Behbehani

Date: 1-23-2026   Lab Section Day: 2(friday)   Lab Section Time: 1-3:50

Integration I: Lab-02
Teensy 4.1 Microcontroller Tutorial

- Each student is **individually responsible** for the work on each lab assignment.
- Review the Lab-02 Discussion Notes, then **carefully** read through the lab assignment.
- Follow the lab instructions **step-by-step**. Use the Discussion Notes as a reference.
- If you have questions, ask your Instructor, TA, or collaborate with your classmates.

Performance Sign-off

[illegible] **Pre-Lab Laptop Setup** [✓ Complete] Incomplete

Comments:

[illegible] **Experiment 1: Teensy LED Blink Sketch**

Comments:

[illegible] **Experiment 2(a): Digital RGB LED Blink**

[illegible] **Experiment 2(b): Analog (Variable Intensity) Red-Green Fade**

Comments:

[illegible] **Experiment 3: Serial Monitor and Pushbutton Interface**

Comments:

[illegible] **Experiment 4(a): Analog Input – Potentiometer**

[illegible] **Experiment 4(b): Analog Input – Temperature or Light Sensor**

Comments:

[illegible] **Experiment 5: Morse Code – "SOS"**

Comments: