Robotics from Scratch: Module 07

Analog, Digital, & Pulse Width Modulation

Summary

1. Digital Signals

- **Concept:** A binary signal. It is either **HIGH (3.3V)** or **LOW (0V)**.
- **Input Example:** A Push Button (Pressed vs. Not Pressed).
- **Output Example:** An LED (On vs. Off).
- **Key Code:** `digitalio.DigitalInOut`

2. Analog Signals

- **Concept:** A continuous range of voltage.
- **Hardware:** We used a **Potentiometer** (variable resistor) to create a voltage divider.
- **The ADC:** The Pico has an **Analog-to-Digital Converter**. It turns voltage (0V–3.3V) into a number (0–65535).
- **Key Code:** `analogio.AnalogIn`

3. PWM: Pulse Width Modulation

- **Problem:** The Pico cannot output 1.5V to make a light dim. It only has 3.3V.
- **Solution:** We turn the power ON and OFF thousands of times per second.
- **Duty Cycle:** The percentage of time the power is ON (0% to 100%).
- **Key Code:** `pwmio.PWMOut`

CircuitPython Syntax Cheat Sheet

```
import board
import time
from digitalio import DigitalInOut, Direction, Pull
import pwmio
from analogio import AnalogIn

# Digital Input (Button)
btn = DigitalInOut(board.GP14)
btn.direction = Direction.INPUT
btn.pull = Pull.UP          # Connects internal resistor to 3.3V

# PWM Output (Fading LED)
led = pwmio.PWMOut(board.GP15, frequency=5000, duty_cycle=0)

# Analog Input (Potentiometer)
knob = AnalogIn(board.GP26) # Must be GP26, 27, or 28

# The Mapping (Knob controls LED)
led.duty_cycle = knob.value
```

Exercises & Challenges

Complete these challenges at home to master the concepts.

Challenge 1: Inverting Potentiometer Logic

Currently, when you turn the knob to the right (high number), the light gets brighter. **Goal:** Change the code so that the light gets dimmer the higher you turn the knob.

- Low Knob Value → Bright Light.
- High Knob Value → Light Off.

Hint: CircuitPython math! If the max value is 65535, how do you flip the number?

Challenge 2: The "Strobe Controller" (Speed Control)

Instead of controlling brightness, use the Potentiometer to control **how fast** the LED blinks on and off.

- Knob Left (Low) → Blinks very fast (or solid on).
- Knob Right (High) → Blinks very slow.

Hint: Use the knob value inside `time.sleep()`. However, if you sleep for 65,535 seconds, you'll be waiting a long time! You need to shrink the number using division.

```
time.sleep(knob.value / 65535)
```

Challenge 3: The "Safe Cracker" (Ranges)

Write a program where the LED is OFF most of the time. It should only turn ON if the knob is in a very specific "sweet spot" in the middle. **Goal:** LED turns ON only if the value is between 30,000 and 35,000. **Hint:** Use the `and` operator.

Challenge 4: The Dead Man's Switch

Combine the Button and the Potentiometer. Write code where the Potentiometer controls the brightness of the LED, but **ONLY** while the button is being held down. If the button is released, the light should turn off.

Topics for Deeper Study

1. **Why 65,535?** Why is that the maximum number for the Analog Input? Search for "16-bit integer max value" or "Binary 2 to the power of 16."
2. **Floating Pins:** In class, we used `Pull.UP` for the button. Try removing that line of code and see what happens when you don't press the button. Does the value stay steady, or does it jitter?
3. **PWM Frequency:** We set our frequency to 5000 Hz. What happens if you change that number to 5 Hz? Try it. What does your eye see? This demonstrates the difference between "dimming" and "blinking."
4. **Switch Debouncing:** When you press a button, two pieces of metal smash together. They don't make perfect contact instantly—they bounce microscopically! Search for "Switch Bounce Oscilloscope" to see what a messy signal looks like and how coders fix it.
5. **The Voltage Divider:** A potentiometer is actually two resistors fighting each other. This circuit is called a "Voltage Divider." Search for the formula. If you have two equal resistors, what voltage comes out the middle? (Hint: It relates to why the knob in the middle reads roughly 32,768).