CEC Robotics

Microcontroller Lab 1 – Blinker Controller

Required Equipment and Supplies

- Raspberry Pi Pico (RP2040 dev. Board)
- USB Micro Cable
- Push Buttons [aka momentary switches] (2)
- 15kΩ Resistors (2)
- LED
- 820Ω Resistor (1)
- Breadboard
- Cables and 22ga wire as needed
- Bench-top DC power supply

Summary

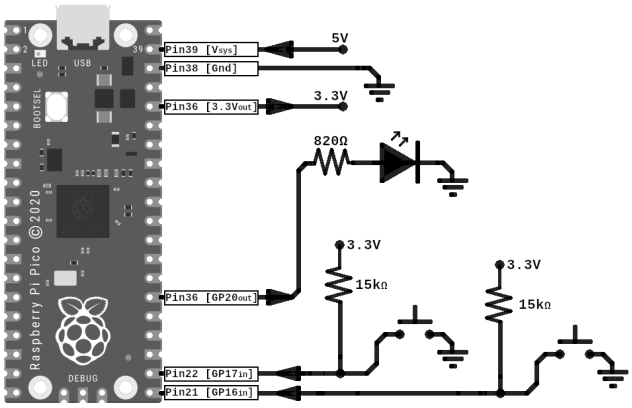
Create a simple microcontroller based device on your breadboard with a blinking external LED and two buttons that control the rate of blinking.

Part I. Circuit Construction

Construct the following circuit as described in the diagram on the right.

Notes

- Important: **Turn off or disconnect V_{sys} from power while programming by USB**.
- **Do not apply more than +6V** by the bench power supply, it may damage the MCU.
- The small → and ← arrows on the wiring are simply to indicate intended inputs vs outputs. They are not devices, but simply indicators of the typical current flow on the wire.
- The 15kΩ resistors are *pull up resistors* which means the **push buttons are active low –** the pin will show 0V when the button is pressed.



Part II. MCU Programming

Create a new code project using the Pi Pico C SDK and Pi Pico Project Template on cec-code-lab.aps.edu/robotics. Use API functions `gpio_init()` and `gpio_set_dir()` to setup the GPIO pins. `gpio_put()` and `gpio_get()` can be used to read and write digital signal voltages. For details on these functions, consult the R. Pi Pico API webpage

Technical Code Objectives

-Write a basic program to blink the external LED.

-Setup the two button pins as inputs. Code logic such that the LED blinks faster when the GP17 button is pressed, and slower when the GP16 button is pressed.

-The LED blink should have a maximum and a minimum speed.

-Create *global constant variables* with 'human readable names' for the GPIO port number corresponding to the LED, faster button and slower button.

-Create and use a *global variable* for the blink delay and *global constant variables* for blink increment and maximum delay. You may use the variable for the delay increment as the minimum or create a minimum delay variable.

-Create a *function* to handle both button inputs named something like `check_rate_buttons()`. Call this function in your primary while loop before making any changes to the LED pin to increase responsiveness.

-Create another function to enclose all GPIO initialization such as `gpio_init()` and `gpio_set_dir()` function calls. Make sure to call this function once before your primary while loop.