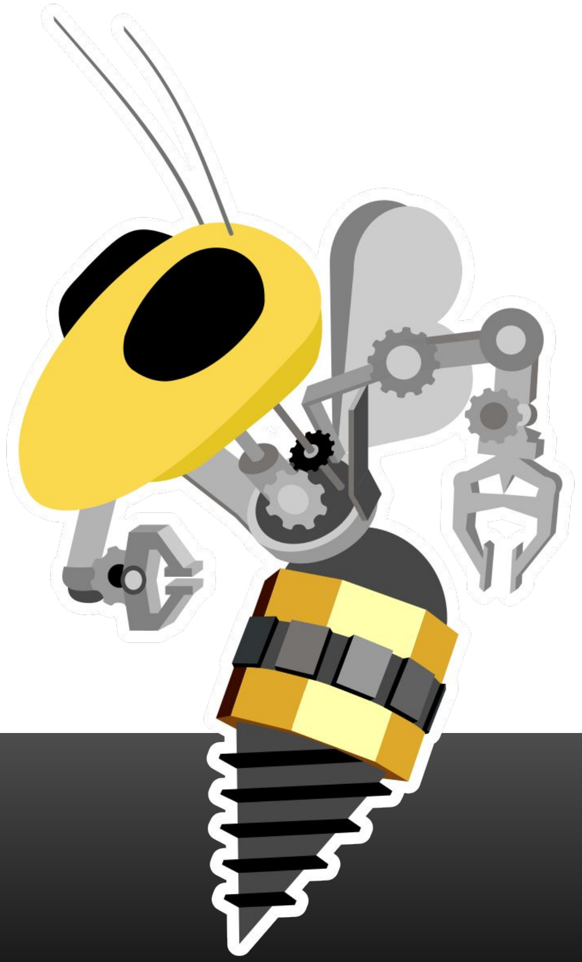
Welcome!

Electrical/Firmware Training

Week 1



RoboJackets

Competitive Robotics at Georgia Tech

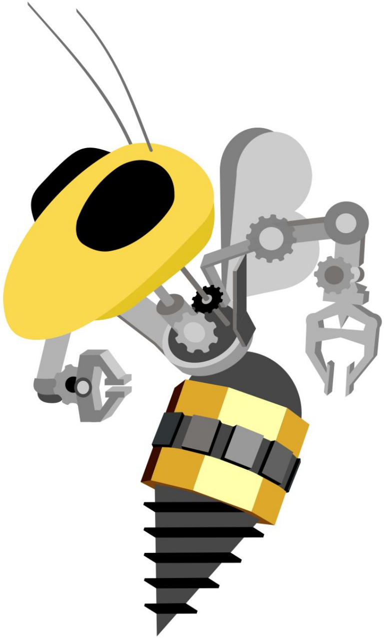
www.robojackets.org

Last Week!

- Introductions
- What is RoboJackets Electrical/Firmware?
- Logistics
- Electrical Basics

This Week!

- Microcontrollers & Firmware
- Arduino, Part 1
- Prototyping

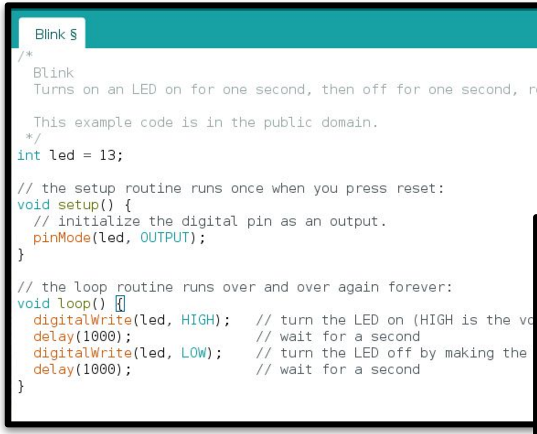


Microcontrollers and Firmware

Arduino!

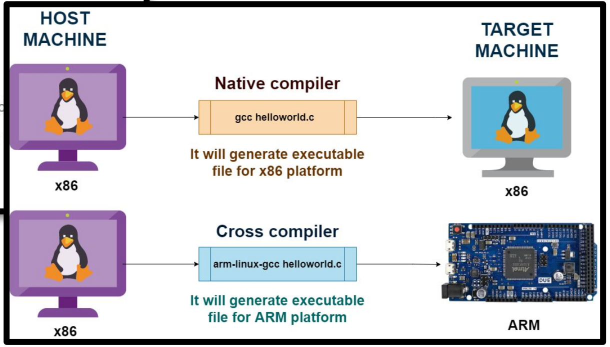
- Arduino is…
  - … a development board (Arduino Uno, Arduino Nano)
  - … a programming language (libraries, compiler, syntax)
- Development board centered around microcontroller

Programming a MCU

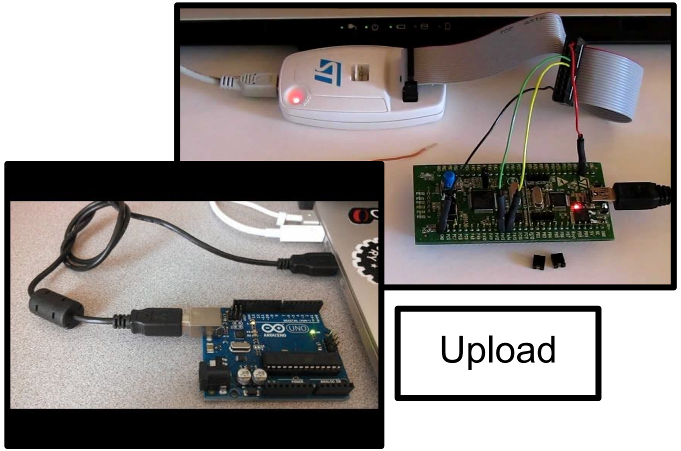


```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the vo
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the
  delay(1000);               // wait for a second
}
```

Code


HOST MACHINE
TARGET MACHINE
Native compiler
gcc helloworld.c
It will generate executable file for x86 platform
x86
x86
Cross compiler
arm-linux-gcc helloworld.c
It will generate executable file for ARM platform
x86
ARM

Cross-compilation



Upload

Example: RoboWrestling

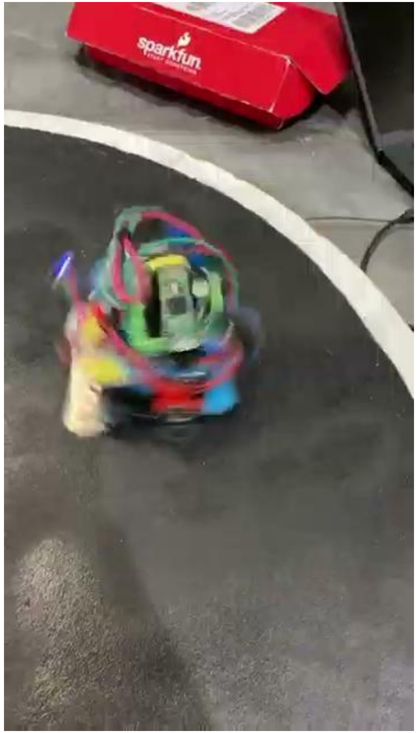
High-level Decisions (Software)

- Where am I in the dohyo (ring)?
- Where is my opponent?
- Where should I move?
- If I made contact with the opponent, what should I do?

Example: RoboWrestling

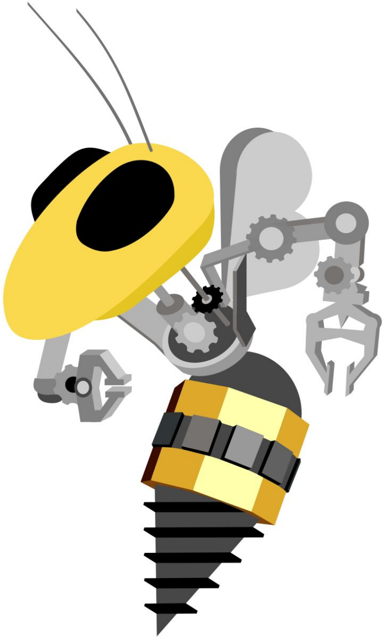
Low-level Decisions (Firmware)

- How do I instruct the motors to spin at the desired speed?
- How do I instruct the robot to start?



Why aren’t Low-level decisions made by a “software computer?”

- Speed
  - Counting encoder ticks <<< Computer vision algorithm
- Space
  - MCUs are small, PCs are not
- Overkill
  - Some robots don’t need to make complicated decisions
  - PC would be $$$ to replace
- Convenience
  - Robotics hardware + software APIs cater to MCUs

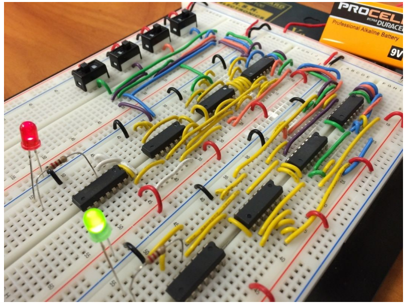


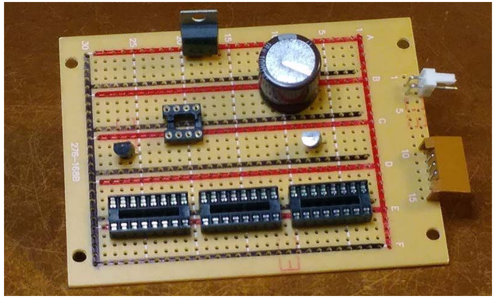
Prototyping

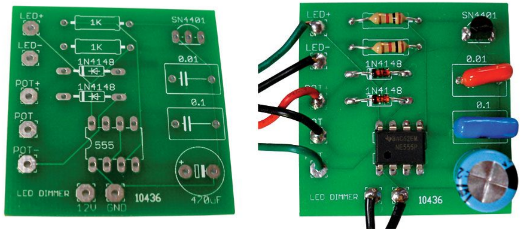
Breadboard and Arduino Uno

From Breadboard to PCB

- Solderless Breadboard (Top)
  - Easy to change
  - Good for initial prototyping
  - Difficult to build large circuits
- Protoboard (Middle)
  - More permanent; solder used
  - Good for mature prototype
  - Changes are doable, but involves re-soldering
- Printed Circuit Board (Bottom)
  - Most permanent; unable to change connections
  - Able to tightly pack components
  - Circuit can span several layers
  - Good for finished product







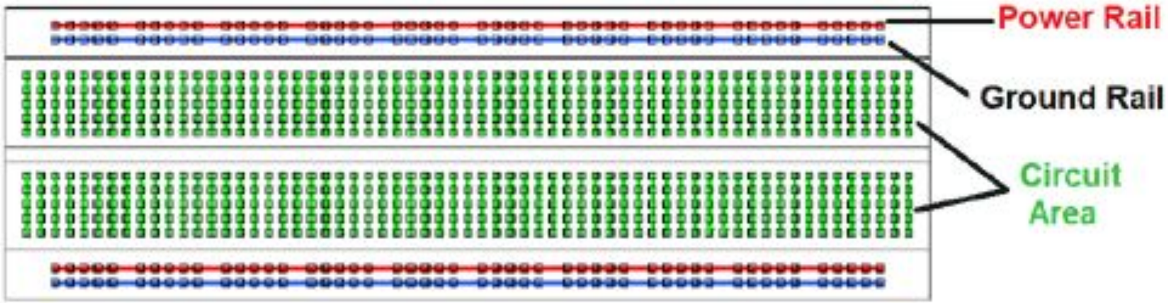
Prototype basic circuit designs with **Breadboards**

Breadboards help you connect electrical components to build basic circuits.

Terminals are the vertical columns. Each terminal is independent from the other

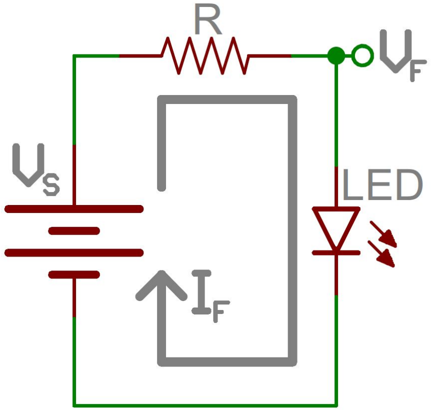
Power rails are use to connect the power supply to the breadboard. The horizontal pins on each power rail are connected.

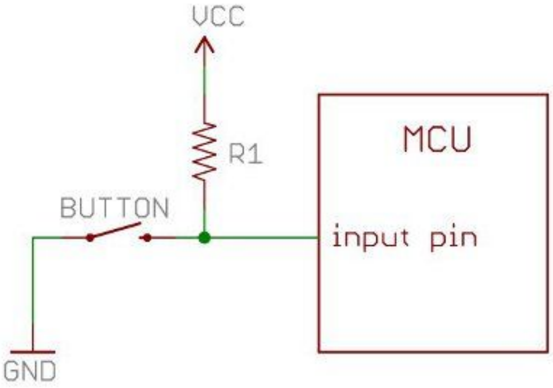
Top half and bottom half of the breadboard are independent from each other.



Prototyping Hardware

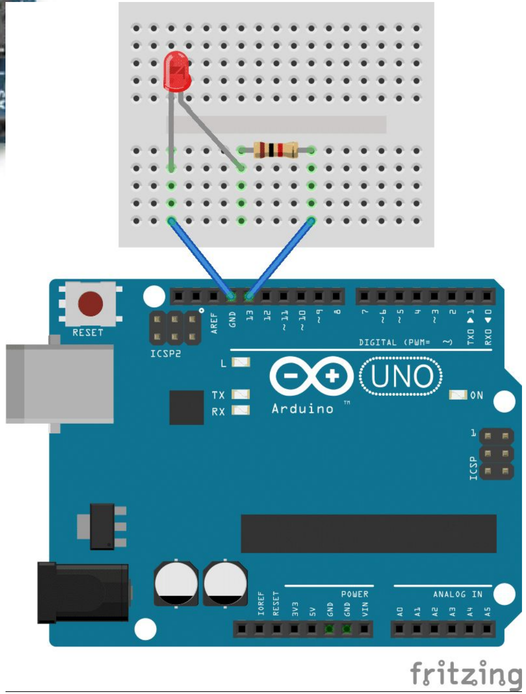
- LEDs
  - Diode: allows lots of current in one direction
  - LED: too much current = blow up
- Buttons
  - Open / close a circuit
  - Problem: High Impedance / “floating” pin
- Resistors
  - Current-limiting resistor + LEDs (I = V / R)
  - Pull-up resistor + Buttons





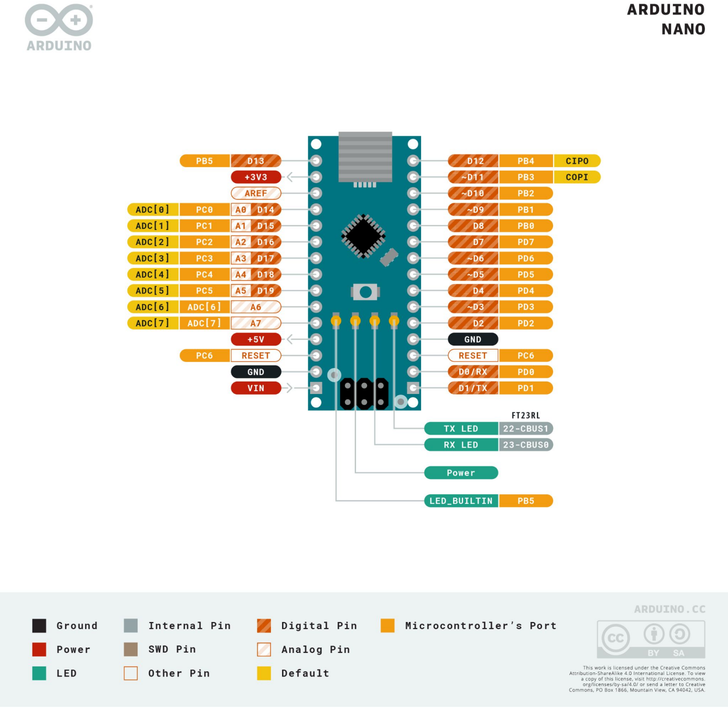
Arduinos!

Microcontroller with I/O ports to control electronics



The pinout diagram

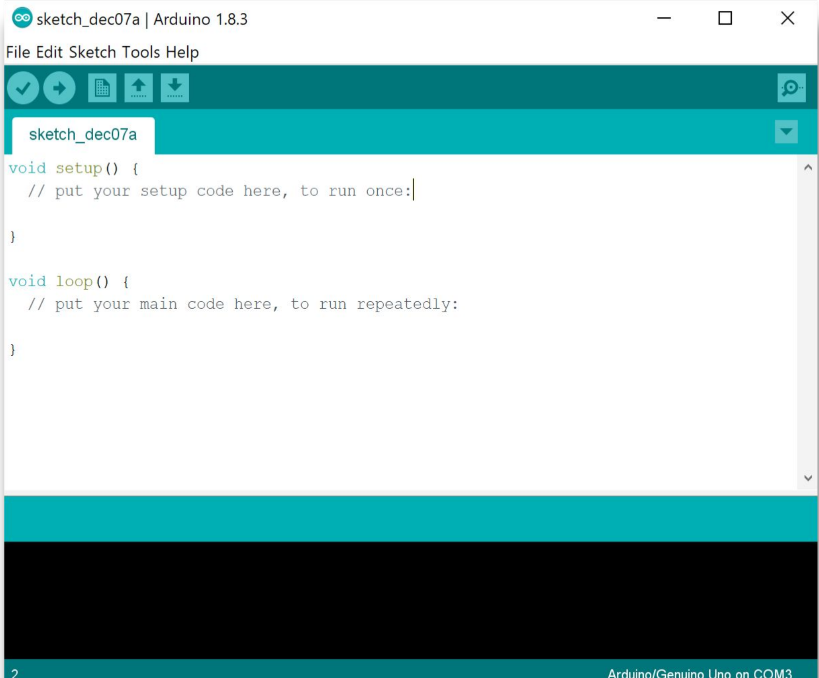
- Arduino Nano Pinout
  - Physical pins -> software pins
    - Ex: D2 (hardware)
    - -> “2” (software)
- Some important pins:
  - PWM#: Digital (0 or 1)
  - A#: Analog (not restricted to 0 or 1)
  - VIN: Input voltage
- If you’re not sure what a pin does, read the datasheet!



Arduino IDE Explained

- IDE = “Integrated Development Environment”
- Programs = “sketches”.

- ✓ **Verify** checks for errors and *compiles* code
- ➔ **Upload** *compiles* and *uploads* code
- **New** creates new sketch
- **Save** saves your sketch
- **Serial Monitor** displays print outputs

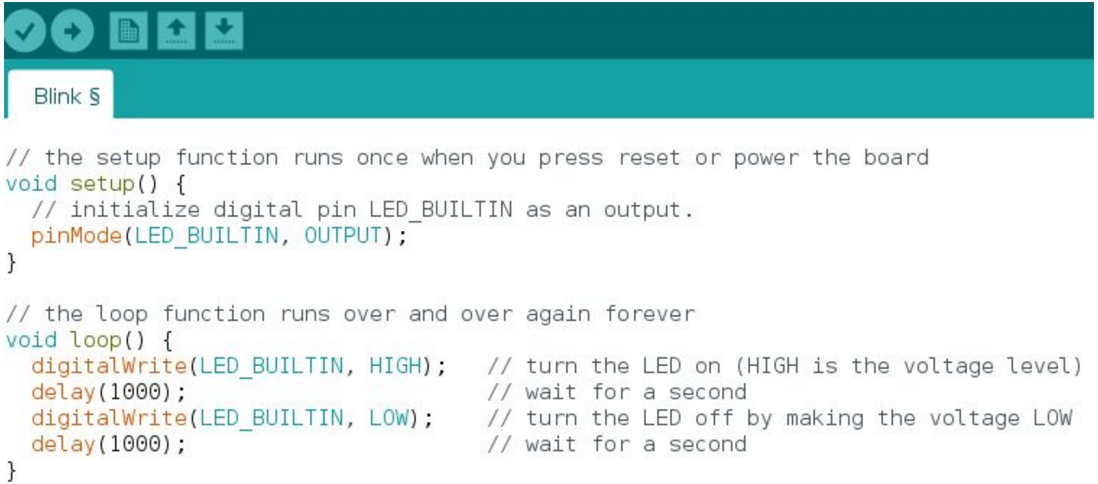

sketch_dec07a | Arduino 1.8.3
File Edit Sketch Tools Help
sketch_dec07a

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

}

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

}
```

2
Arduino/Genuino Uno on COM3

Blink Example

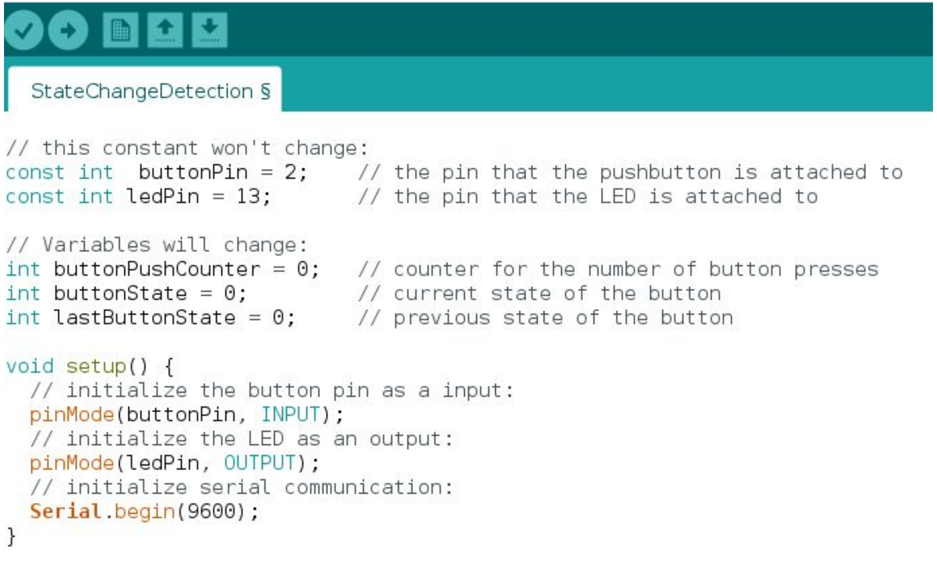
- setup()
- loop()
- pinMode()
- digitalWrite()
- delay()



```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

StateChangeDetection, Part 1

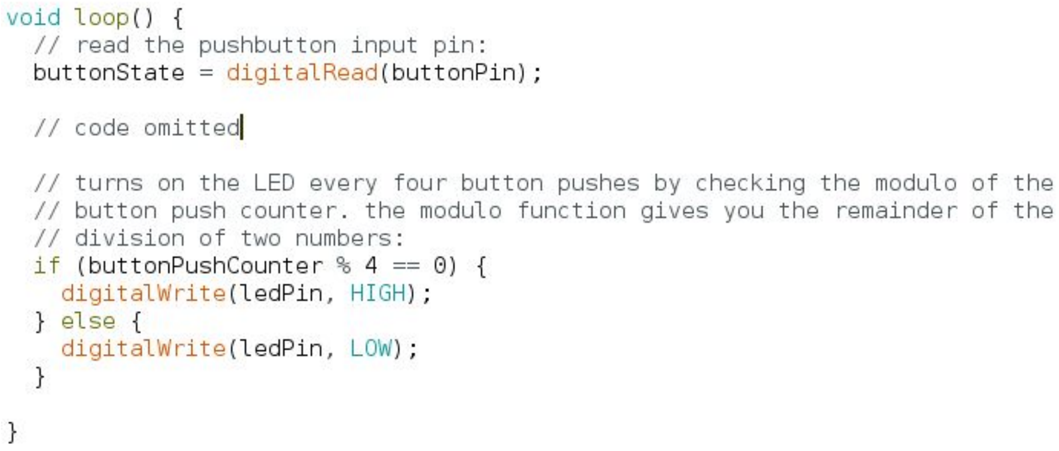
- Variables
  - Pin variables (buttonPin, ledPin)
  - Integer variables (buttonPushCounter, buttonState, lastButtonState)
- pinMode()
- Serial.begin()



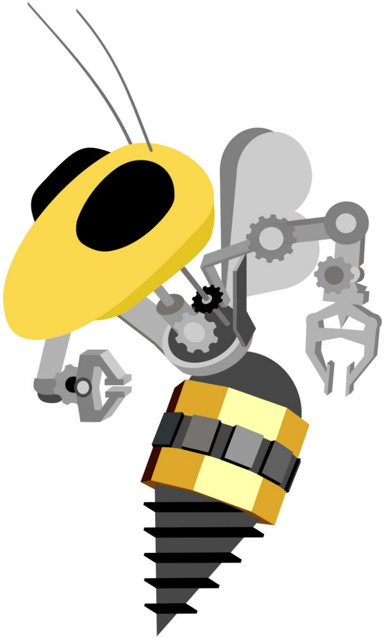
```
// this constant won't change:
const int  buttonPin = 2;    // the pin that the pushbutton is attached to
const int ledPin = 13;       // the pin that the LED is attached to

// Variables will change:
int buttonPushCounter = 0;   // counter for the number of button presses
int buttonState = 0;         // current state of the button
int lastButtonState = 0;     // previous state of the button

void setup() {
  // initialize the button pin as a input:
  pinMode(buttonPin, INPUT);
  // initialize the LED as an output:
  pinMode(ledPin, OUTPUT);
  // initialize serial communication:
  Serial.begin(9600);
}
```

StateChangeDetection, Part 2

- digitalRead()
- If /else
- digitalWrite()



```
void loop() {
  // read the pushbutton input pin:
  buttonState = digitalRead(buttonPin);

  // code omitted

  // turns on the LED every four button pushes by checking the modulo of the
  // button push counter. the modulo function gives you the remainder of the
  // division of two numbers:
  if (buttonPushCounter % 4 == 0) {
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }

}
```

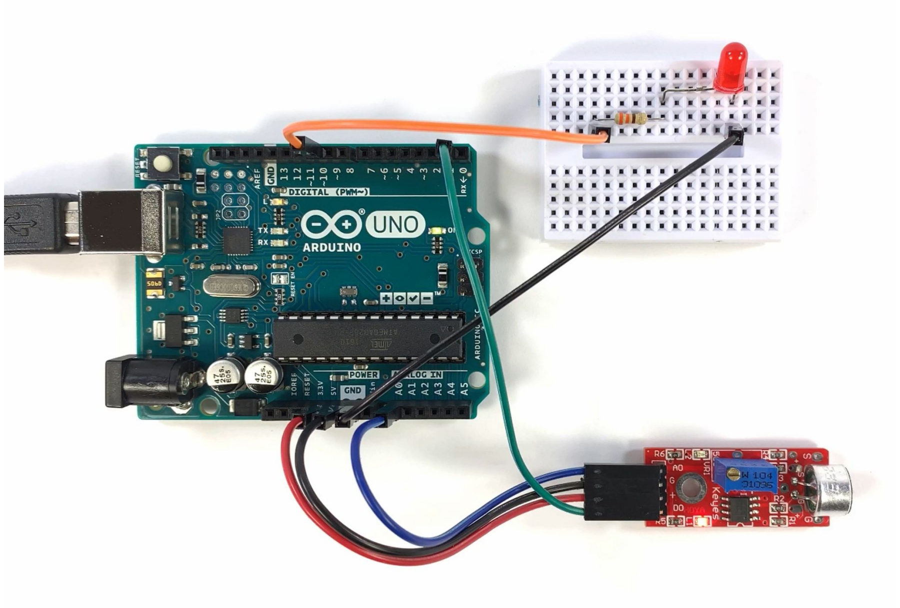


Lab

LEDs and Buttons

Lab Time!

- Read the lab setup
  - Legacy Arduino IDE
  - CH340 drivers
- Challenges!
  - Blink
  - Blink + Button
  - Variable Frequency Blink + Button
  - Binary Counter



For next time...

Bring the following:

- Laptop
- Mouse (highly recommended unless you’re a menace with trackpad)
- A Coke for Kyle

Location:

- Firmware: Skiles 255 (Monday) and Van Leer 457 (Friday)
- Electrical: Skiles 169 (Monday and Friday)

Feedback