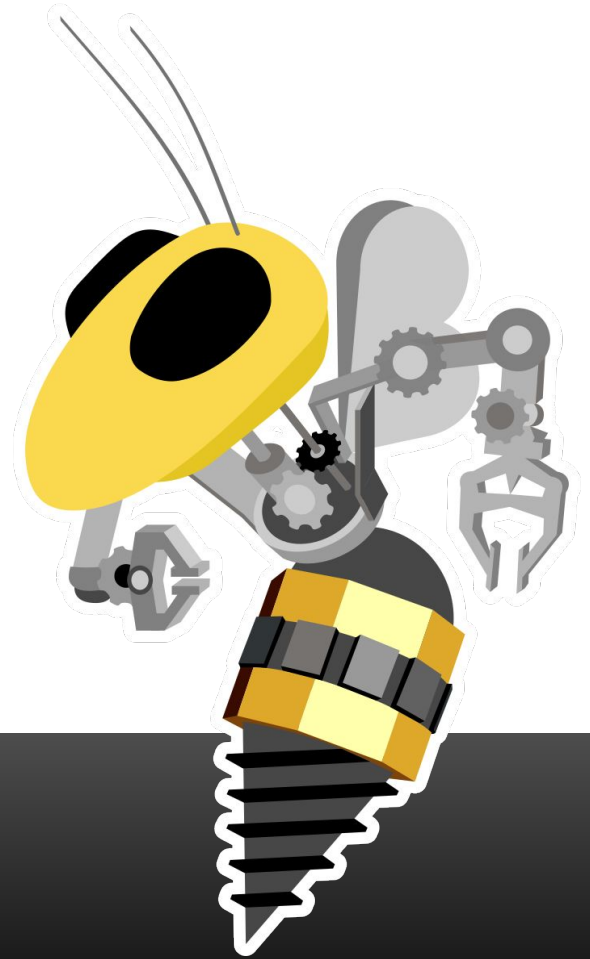


# Welcome!

Electrical/Firmware Training  
Week 1

**ROBOJACKETS**  
COMPETITIVE ROBOTICS AT GEORGIA TECH

*[www.robojackets.org](http://www.robojackets.org)*

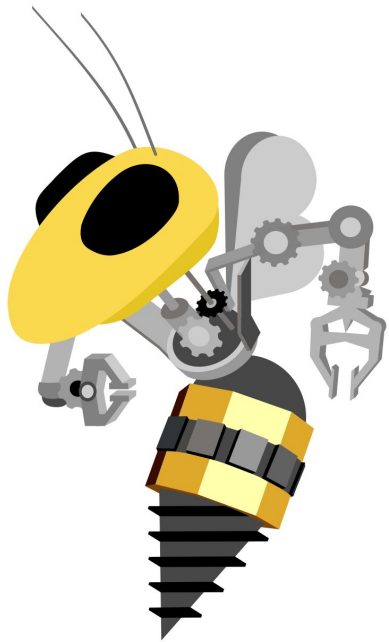


# Last Week

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

# Agenda

1. Microcontrollers (MCU)
2. Intro to C++
3. Prototyping
4. Lab

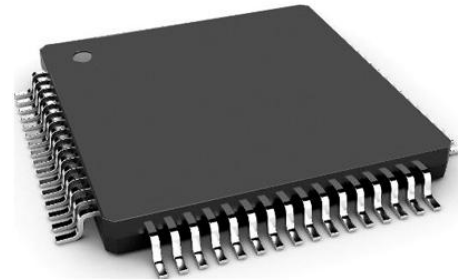


# Microcontrollers

aka MCU

# What is a Microcontroller?

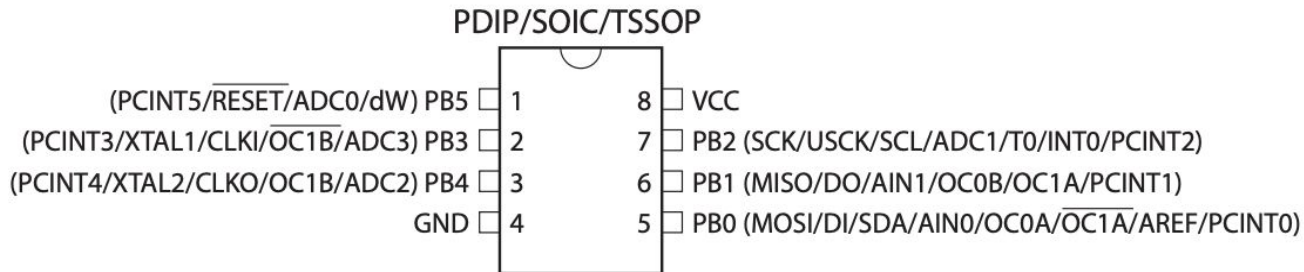
- The microcontroller is like a brain
- It's a computer on a single electronic chip
- Contains logic gates, capacitors, resistors, etc.
- Functions as processor, memory, programmable input/output peripherals
- These have pins for power, communication, and input/output connections



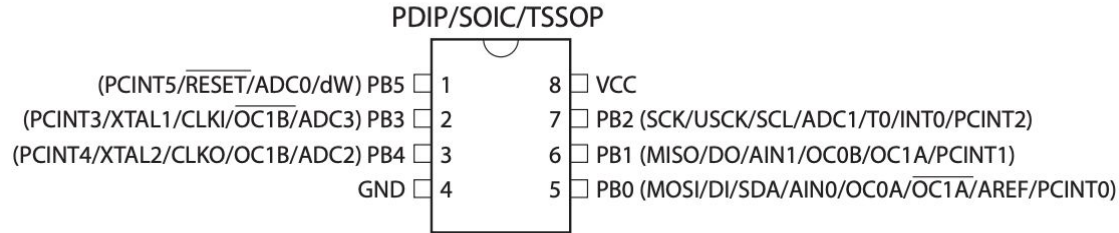
# Pin config for ATtiny25

*Pin configurations help to identify inputs and outputs for the microcontroller*

- A pin is a piece of metal that sticks out in an microcontroller that can be connected electrically to the outside world
- Each pin serves different functions, but not all pins need to be connected to something
- The MCU below is an 8-pin microcontroller



# Pin config for ATtiny25



NOTE: TSSOP only for ATtiny45/V

- **VCC** is used to power the microcontroller
- **GND** is used to connect the MCU to ground
- **RESET** resets the program when its input is *low*
- **CLK/SCK** govern the speed at which the processor executes instructions
- **MOSI** and **MISO** let the microcontroller communicate with peripherals by sending and receiving data
- **ADC** is an analog to digital converter

# Intro to C++

*A programming  
language*

C++ is a programming language  
respected for its flexibility and low-level  
functionality.



Used in operating systems, microcontrollers, and  
elsewhere. C++ files have a .cpp extension.

Like most programming languages, code runs from  
top to bottom. Before the code is sent to a computer,  
it is **compiled** into binary code and then **executed**.

More about this in Week 2 of firmware



# C++ Variables & Types

- int → integers (e.g. 12, -17)
- double → floating point or decimal numbers (e.g. 1.23)
- char → single characters (e.g. "s", "D")
- string → text (e.g. "Hello world")
- bool → logical values (e.g. true/false)

```
//set up and  
//assign a  
//value like below  
int i = 5;  
//you can also set  
//up a variable  
//without initially  
//assigning a value  
bool rain;  
rain = false;
```

# C++ Arithmetic Operators

- $+$   $\rightarrow$  Addition
- $-$   $\rightarrow$  Subtraction
- $*$   $\rightarrow$  Multiplication
- $/$   $\rightarrow$  Division
- $\%$   $\rightarrow$  Modulo (produces the remainder e.g.  $9 \% 4 \gg 1$ )

# C++ Relational Operations

- `==` → Equal to
- `!=` → Not equal to
- `>` → Greater than
- `<` → Less than
- `>=` → Greater than or equal to
- `<=` → Less than or equal to

# C++ Conditions

- If the condition (within brackets) in the `if` statement is `true`, then the code within the first set of braces will execute and will output "Three!".
- `Else if` statements can be added in between `if` and `else` to provide additional conditions

```
int  num = 4;
if  (num == 3) {
    return "Three!";
} else if (num == 5) {
    return "Five!";
}
```

\*assigning variable\*

\*is equal to\*

Notice the indentation

Notice the braces

# C++ Conditions

- The `else` statement will execute when all previous conditions evaluate `false`
- Note that once a condition is met, no further code within the `if` block will execute

```
int num = 4;
if (num == 3) {
    return "Three!";
} else if (num == 5) {
    return "Five!";
} else {
    return num;
}
```

*\*assigning variable\**

*\*is equal to\**

*Notice the indentation*

*Notice the braces*



# Prototyping

Breadboard and Arduino Uno

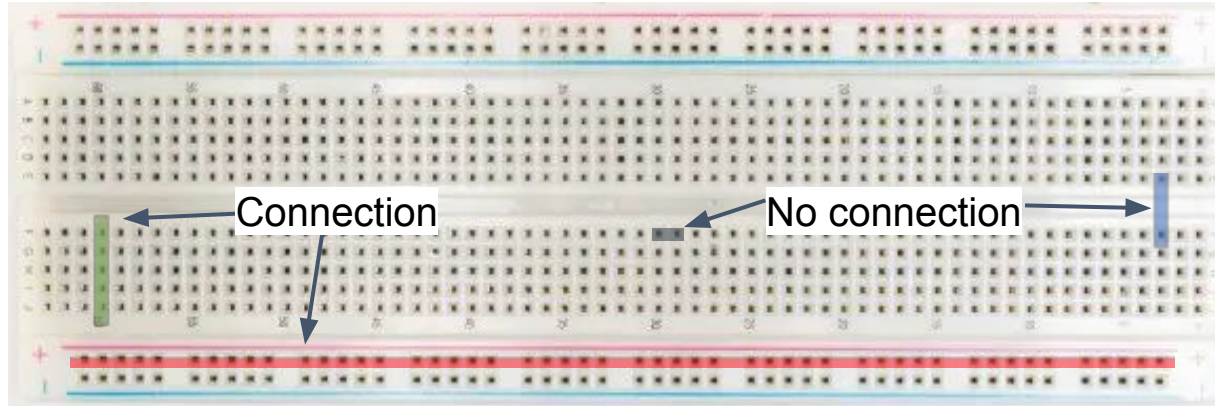
# Prototype basic circuit designs with **Breadboards**

**Breadboards** help you connect electrical components to build basic circuits.

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

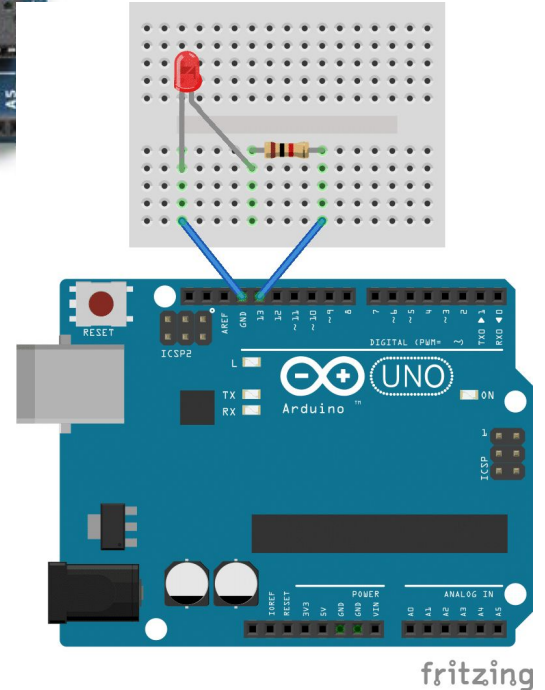
**Power rails** are use to connect the power supply to the breadboard. The horizontal pins on each power rail are connected.

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



# Arduino Uno

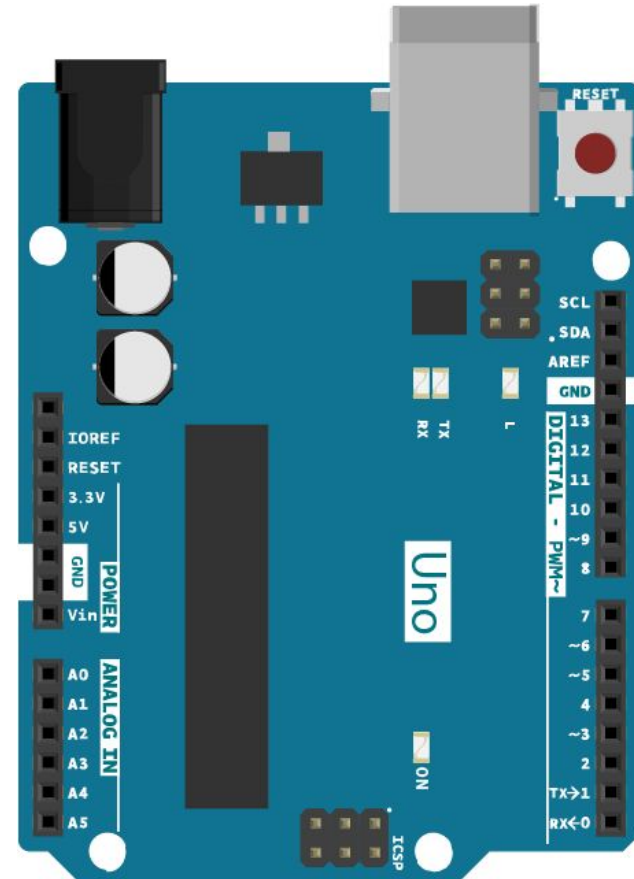
*Microcontroller  
with I/O ports to  
control electronics*





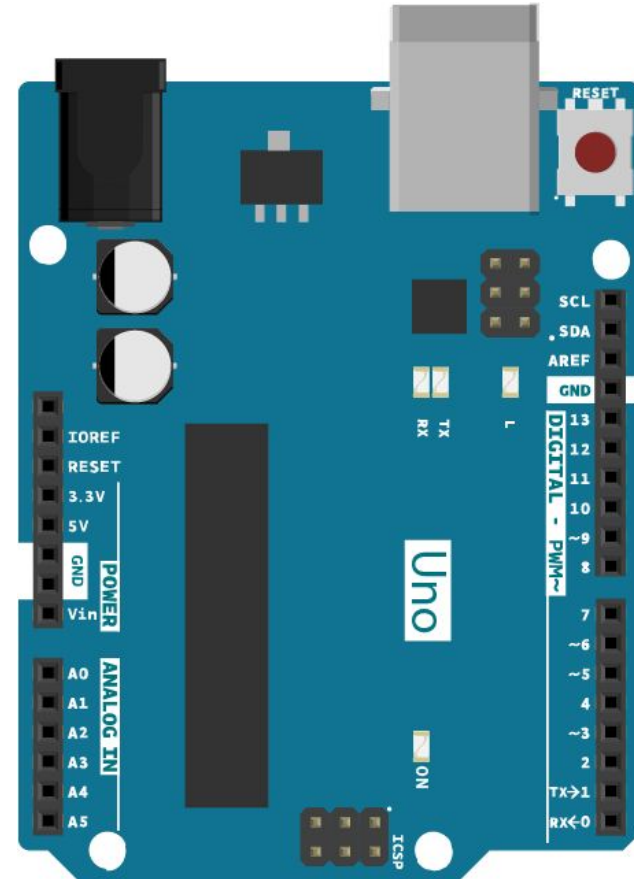
# Arduino Board Explained

- **Power:** an Arduino can be powered using a USB cable or a Barrel Jack. Use between 6V and 12V
- **GND** can connect components to Ground
- **5V & 3.3V:** supplies power to components
- **ANALOG IN:** A0 - A5 pins can read values from analog sensors
- **DIGITAL** pins can be used as input from components (like switches) or output to components (like LEDs)
- **PWM:** some pins have a tilde (~) symbol next to it for Pulse Width Modulation (PWM) to simulate analog output.



# Arduino Board Explained

- **AREF:** used to set external reference voltage between 0V and 5V for analog input. You would mostly ignore this.
- **RESET** (button): temporarily connects reset pin to ground to restart code. Useful when you cannot reset with a computer.
- **Built-in LED:** a mounted LED that can be programmed to blink.



# Arduino IDE Explained


- IDE aka integrated development environment
- Programs written using Arduino Software (IDE) are called **sketches**.

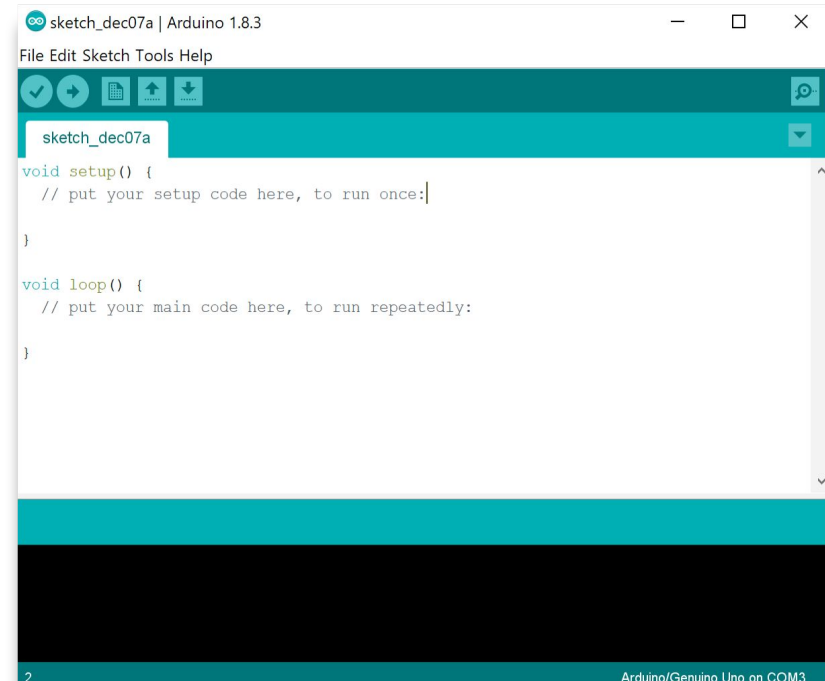
 **Verify** checks for errors and *compiles* code

 **Upload** *compiles* and *uploads* code

 **New** creates new sketch

 **Save** saves your sketch

 **Serial Monitor** displays print outputs



# Arduino IDE Explained

- `void setup()`
  - *initialize* variables and *define* pins
  - Runs once
- `void loop()`
  - write code to read from and write to pins
  - Runs multiple times
- The Arduino IDE has a lot of built-in *libraries* that boils down complex tasks into simple functions. You can add libraries to your sketch using `#include`
  - **Sketch > Import Library**
- The code used in Arduino IDE is very similar to C++

sketch\_dec07a | Arduino 1.8.3

File Edit Sketch Tools Help



sketch\_dec07a

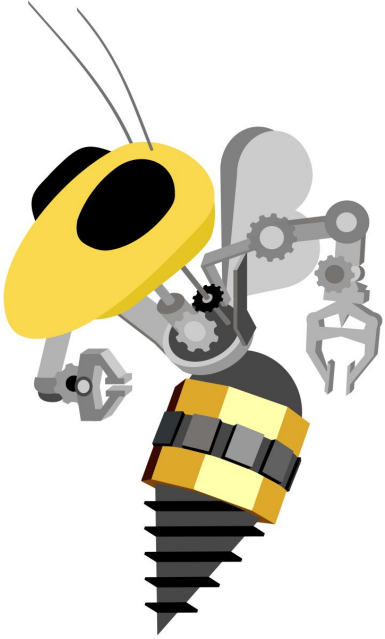
```
void setup() {  
    // put your setup code here, to run once:  
}  
  
void loop() {  
    // put your main code here, to run repeatedly:  
}
```

# Common Arduino IDE Functions

- `pinMode(pin, mode);` *//goes in void setup()*
  - Configures the specified pin to act as an input or output
- `digitalWrite(pin, value);` *//goes in void loop()*
  - Write a high (5V) or low (0V) value to a specified pin
- `digitalRead(pin);` *//goes in void loop()*
  - Read a high or low value from a specified pin
  - Returns true if voltage is HIGH, or false if it is LOW
- `delay(value);` *//goes in void loop()*
  - Pauses code execution for a value—time in milliseconds

# Common Arduino Functions

- `analogRead(pin);` *//goes in void loop()*
  - Returns an analog value from an analog pin
- `Serial.print();` *//goes in void loop()*
  - Used to print values in serial monitor and to see if your code is being executed → Useful for debugging.
  - Requires `Serial.begin(value);` in void setup



# Lab

## LEDs and Buttons

# Lab Setup

This week we will build a simple circuit to light up LEDs and we will use an Arduino Nano to control which LEDs light up.