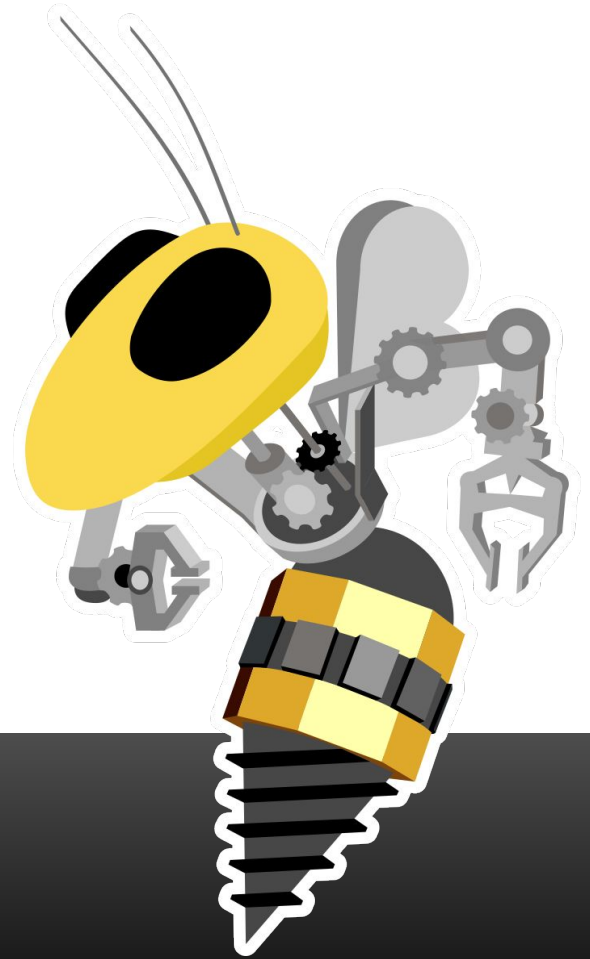


Welcome!

Electrical/Firmware Training
Week 1

ROBOJACKETS
COMPETITIVE ROBOTICS AT GEORGIA TECH

www.robojackets.org

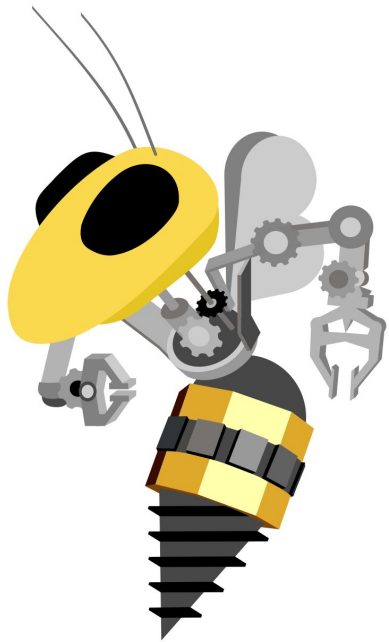


Last Week!

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

This Week!

- Microcontrollers
- Why Firmware?
- C++, Part 1
 - Variables in C++
 - Arithmetic & Making Comparisons
 - If / Else
- Prototyping



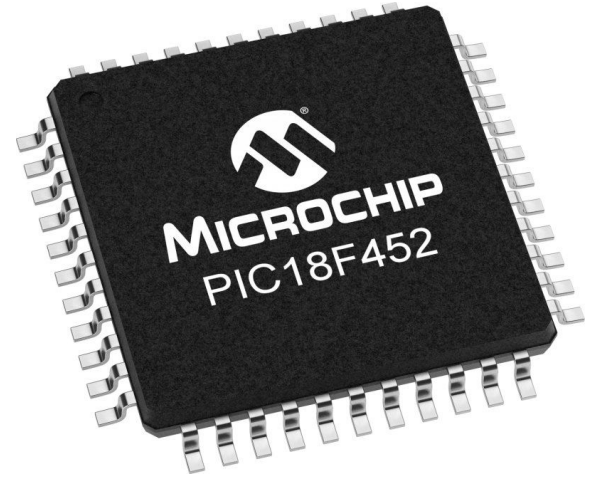
Microcontrollers

aka MCU

What is a Computer?



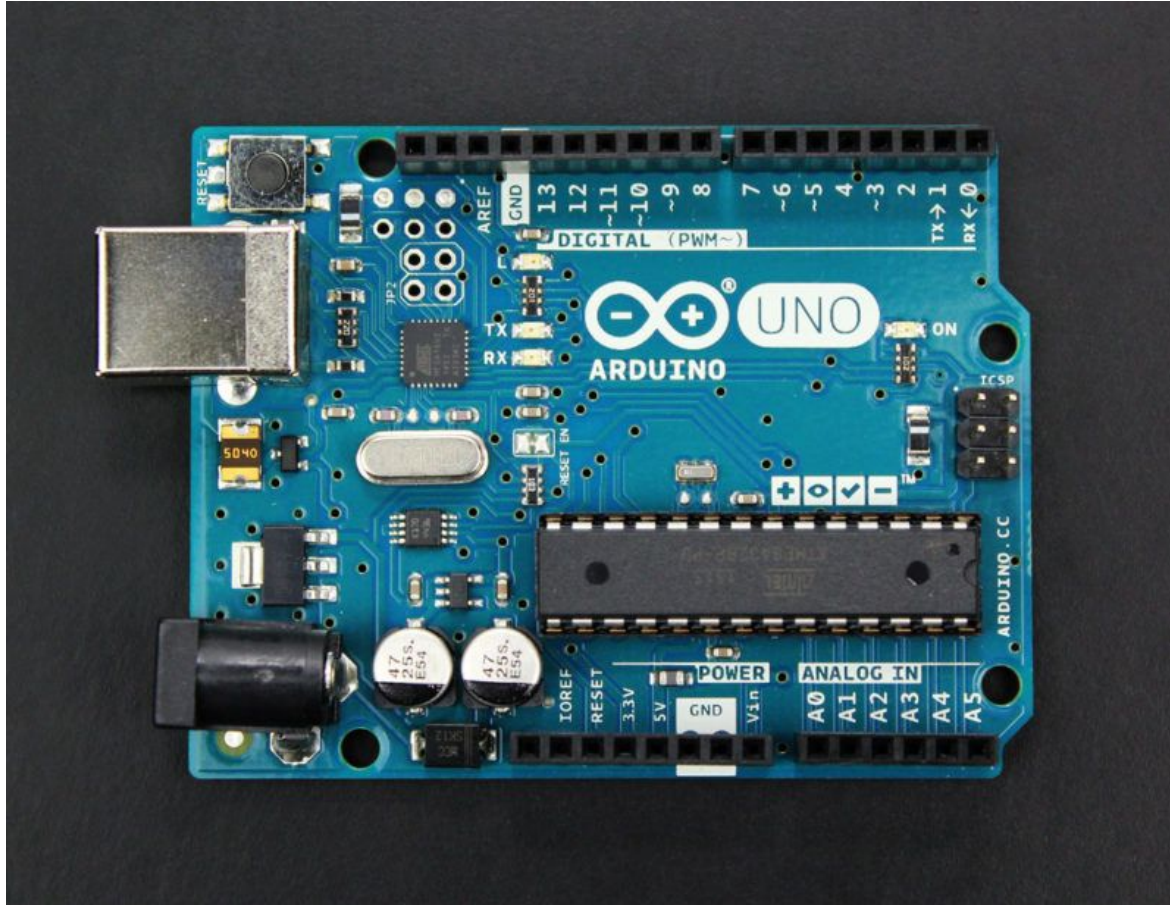
What is a Microcontroller (MCU)?



Website: <https://www.microchip.com/en-us/product/PIC18F452#>

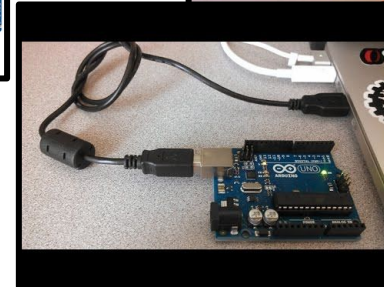
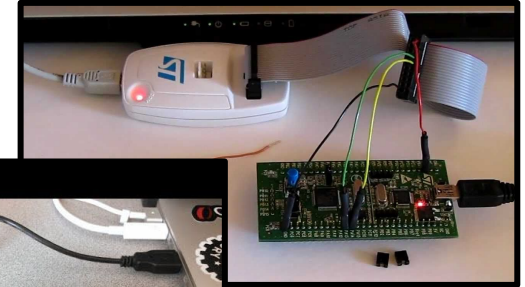
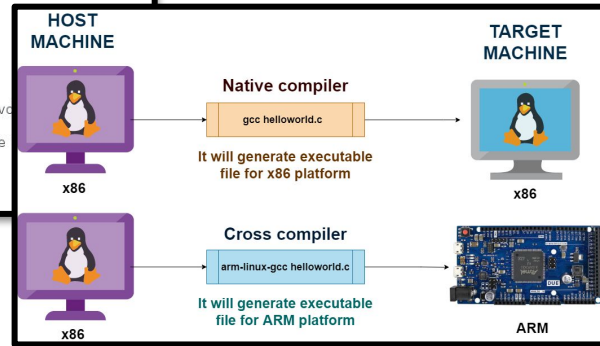
Datasheet:

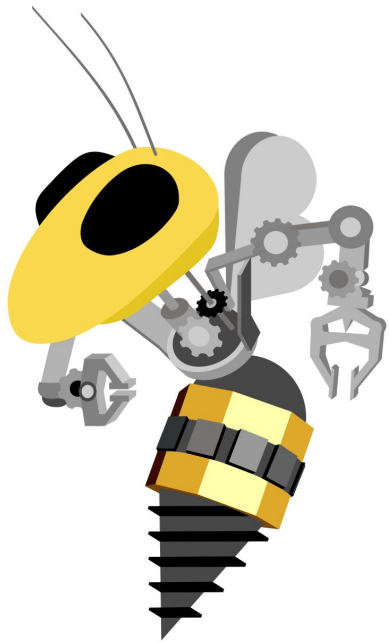
<https://ww1.microchip.com/downloads/aemDocuments/documents/MCU08/ProductDocuments/DataSheets/39564c.pdf>



Programming a MCU

```
Blink §  
/*  
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 voltage level)  
  delay(1000);             // wait for a second  
  digitalWrite(led, LOW);  // turn the LED off by making the voltage LOW  
  delay(1000);             // wait for a second  
}
```





**Why do we need
firmware?**

Firmware

From Wikipedia, the free encyclopedia

In [computing](#), **firmware** is a specific class of [computer software](#) that provides the [low-level control](#) for a device's specific [hardware](#). Firmware, such as the [BIOS](#) of a personal computer, may contain basic functions of a device, and may provide [hardware abstraction](#) services to higher-level software such as [operating systems](#). For less complex devices, firmware may act as the device's complete [operating system](#), performing all control, monitoring and data manipulation functions. Typical examples of devices containing firmware are [embedded systems](#) (running [embedded software](#)), home and personal-use appliances, computers, and [computer peripherals](#).

Firmware is held in [non-volatile memory](#) devices such as [ROM](#), [EPROM](#), [EEPROM](#), and [Flash memory](#). Updating firmware requires ROM [integrated circuits](#) to be physically replaced, or EPROM or flash memory to be reprogrammed through a special procedure.^[1] Some firmware memory devices are permanently installed and cannot be changed after manufacture. Common reasons for updating firmware include fixing bugs or adding features to the device.

Example: RoboCup

The screenshot displays the RoboJackets - Soccer software interface. The central area shows a green soccer field with white lines and goals. Robots are represented by colored circles: blue for the Blue Team and yellow for the Yellow Team. The game is in the "Normal First Half Prep" stage, with a "Force Start" command and 0.00 s left. The score is 0-0, and there are no cards or timeouts recorded. The robot status panel on the left lists four robots (IDs 0, 2, 3, 4) all of type "RJ2008". The behavior tree on the right shows a "TestDefense::running" node, which branches into "Defense::running", "SubmissiveGoalie::block", "Move::completed[robot=5]", "face(-1.56316, 2.70279)", "SubmissiveDefender::marking", "Move::completed[robot=3]", "face(-1.56316, 2.70279)", "SubmissiveDefender::marking", "Move::completed[robot=4]", and "face(1.75555, 1.45242)". The plays panel on the right lists various plays and their scores, including "OurGoalKick", "OurKickoff", "TheirKickoff", "TheirRestart", "Stopped", "testing", "DebugWindowEvaluator", "TestDefense", "LineUp", "RepeatedLineUp", "StressTest", and "TestBump". The status bar at the bottom indicates "TestDefense" is active, "Not Recording", "View: 28.2 fps", "Proc: 59.3 fps", and "Log: 4992/100000 43427 KiB".

RoboJackets - Soccer

Goalie: Manual: 904.0MHz
5 None YELLOW STATUS

Stage Normal First Half Prep
Command Force Start
Time Left 0.00 s

<Yellow Team> <Blue Team>
Score 0 Score 0
Red Cards 0 Red Cards 0
Yellow Cards 0 Yellow Cards 0
Timeouts Left 0 Timeouts Left 0
Goalie ID 0 Goalie ID 0

Robot Status

0 ID: ??-?? RJ2008
2 ID: ??-?? RJ2008
3 ID: ??-?? RJ2008
4 ID: ??-?? RJ2008

Behavior Tree

```
TestDefense::running
  Defense::running
    SubmissiveGoalie::block
      Move::completed[robot=5]
        face(-1.56316, 2.70279)
      SubmissiveDefender::marking
        Move::completed[robot=3]
          face(-1.56316, 2.70279)
        SubmissiveDefender::marking
          Move::completed[robot=4]
            face(1.75555, 1.45242)
```

Plays / Behaviors Config Tree Layers Joystick

Plays

Play	Score
<input type="checkbox"/> OurGoalKick	inf
<input type="checkbox"/> OurKickoff	inf
<input type="checkbox"/> TheirKickoff	inf
<input type="checkbox"/> TheirRestart	inf
<input type="checkbox"/> Stopped	inf
▼ testing	
<input type="checkbox"/> DebugWindowEvaluator	10
<input checked="" type="checkbox"/> TestDefense	10
<input type="checkbox"/> LineUp	10
<input type="checkbox"/> RepeatedLineUp	10
<input type="checkbox"/> StressTest	10
<input type="checkbox"/> TestBump	10

TestDefense Not Recording View: 28.2 fps Proc: 59.3 fps Log: 4992/100000 43427 KiB

High-level Decisions

- Where am I in the world?
- Where are my teammates / opponents / goals?
- Where should I move?
- If I have the ball, when and where should I kick it and how hard?

Low-level Decisions

- How do I instruct the motors to spin at the desired speed?
- How do I instruct the kicker to kick at a certain time, in a certain direction, with the desired intensity?

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



**What should I choose?
Electrical or Firmware?**

Electrical

Circuit Design

- Competition requires X
- Solution: This motor / sensor / indicator / communication module solves X!
- What circuit is needed to support the solution?
- Circuit schematic!

Electrical

Circuit Design

PCB Design (EAGLE CAD)

- Circuit Schematic → PCB

Electrical

Circuit Design

PCB Design (EAGLE CAD)

Physical Electrical Work

- Crimping
- Soldering
- Cutting
- Heat Shrinking
- Making replacement parts

Firmware

- (1) What are inputs / outputs to MCU?
- (2) What peripherals does my MCU need to send / receive data?
- (3) Do I need to transform the data?
- (4) Circuit prototyping
- (5) How do I program the MCU to handle inputs / outputs?
- (6) How does the MCU interact with software?

Intro to C++

*A programming
language*



C is more readable assembly...
C++ is an object-oriented hack of C.

– imprecise quotation of Jim Rehg

<https://github.com/RoboJackets/firmware-training>

Go to Google Colab link at bottom of page!

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!";
}
```

Notice the indentation

Notice the braces

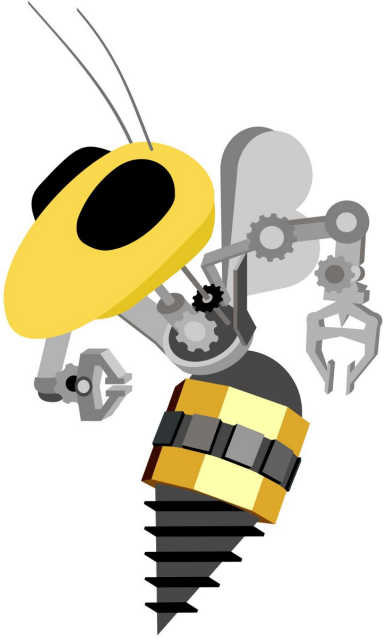
C++ Conditions

- 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;
}
```

The diagram illustrates the execution of the provided C++ code. Annotations include:

- *assigning variable***: Points to the assignment `int num = 4;`.
- *is equal to***: Points to the equality operator `==` in the condition `num == 3`.
- Notice the indentation**: Points to the opening curly brace of the `if` block.
- Notice the braces**: Points to the closing curly brace of the `else` block.



Prototyping

Breadboard and Arduino Uno

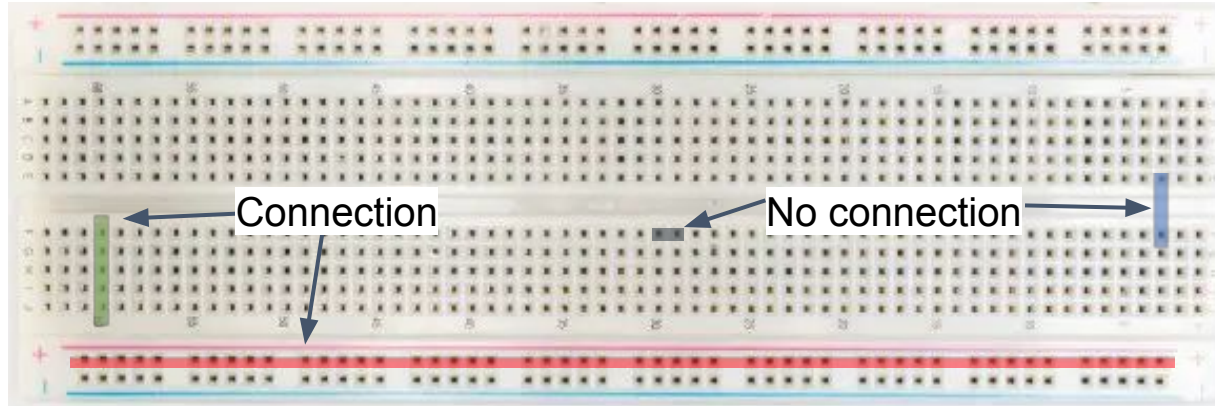
Prototype basic circuit designs with **Breadboards**

Breadboards help you connect electrical components to build basic circuits.

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

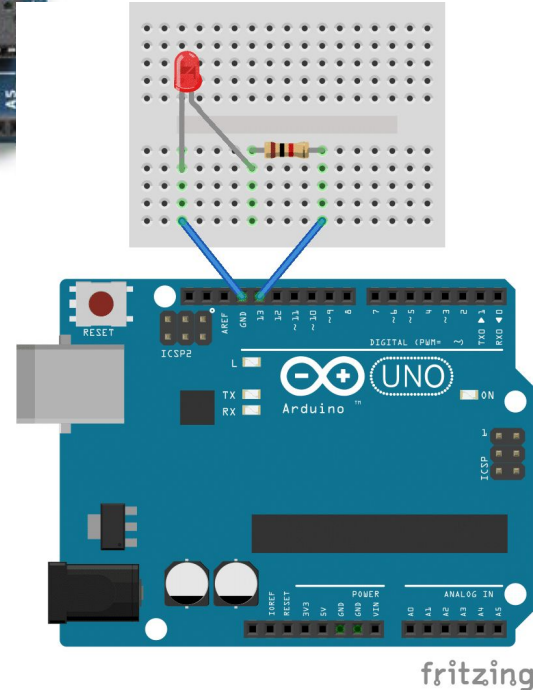
Power rails are use to connect the power supply to the breadboard. The horizontal pins on each power rail are connected.

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



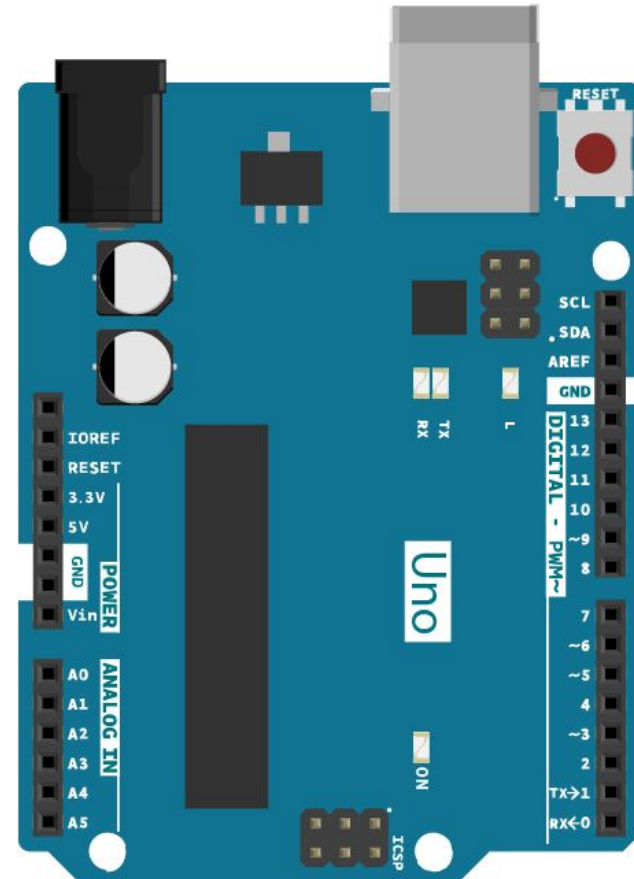
Arduino Uno

*Microcontroller
with I/O ports to
control electronics*



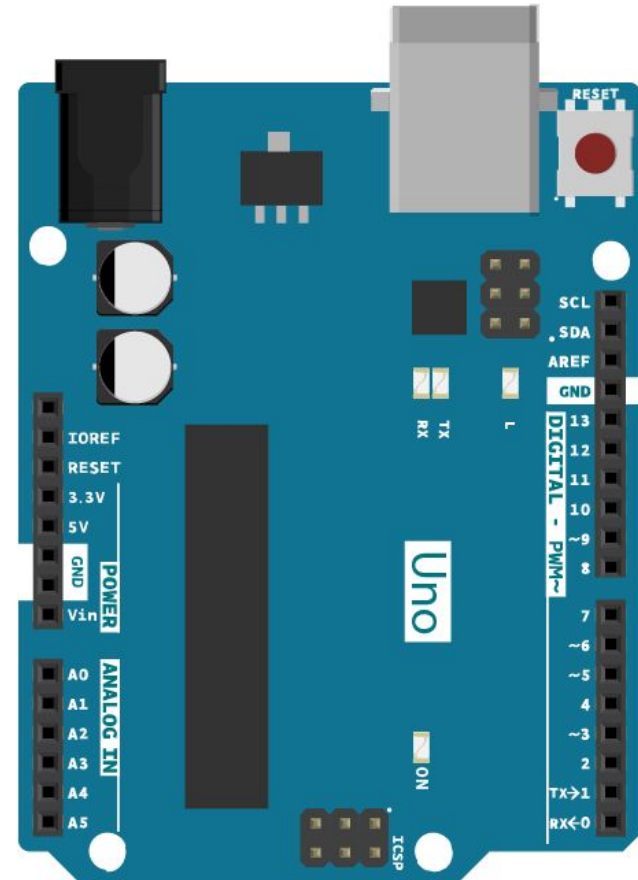
Arduino Board Explained

- **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**.

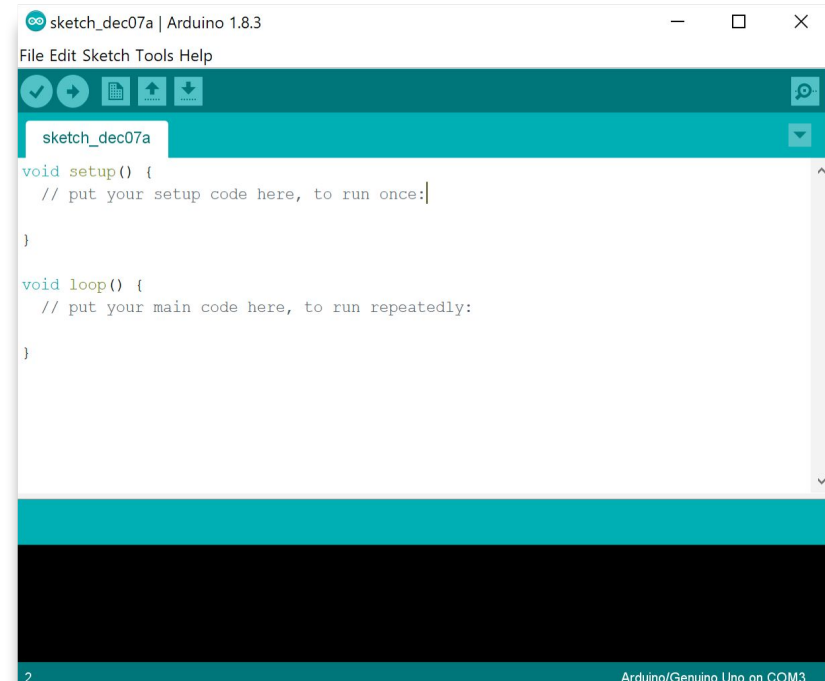
 **Verify** checks for errors and *compiles* code

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

 **New** creates new sketch

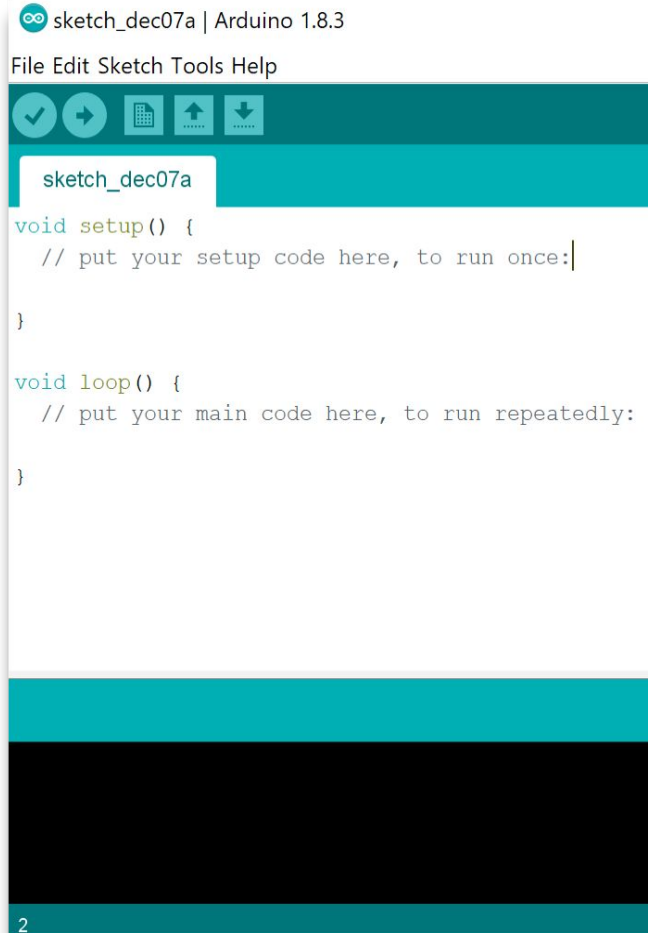
 **Save** saves your sketch

 **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++

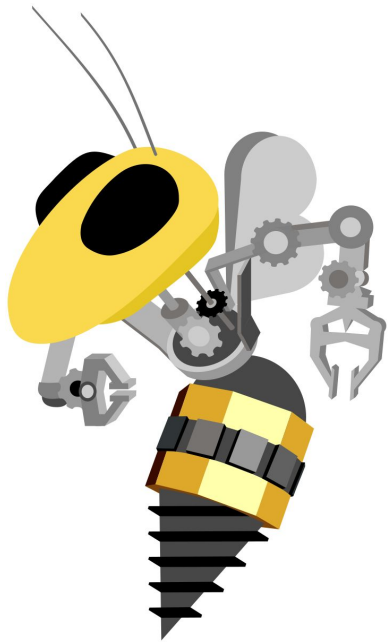


Common Arduino IDE Functions

- `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.