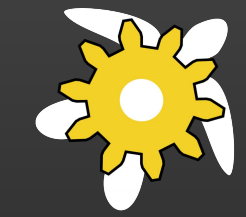
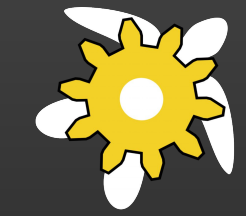


TEXAS A&M UNIVERSITY

ROBOTICS TEAM & LEADERSHIP EXPERIENCE

Design Review and C++

TURTLE Hatchling



TEXAS A&M UNIVERSITY

ROBOTICS TEAM & LEADERSHIP EXPERIENCE

C++

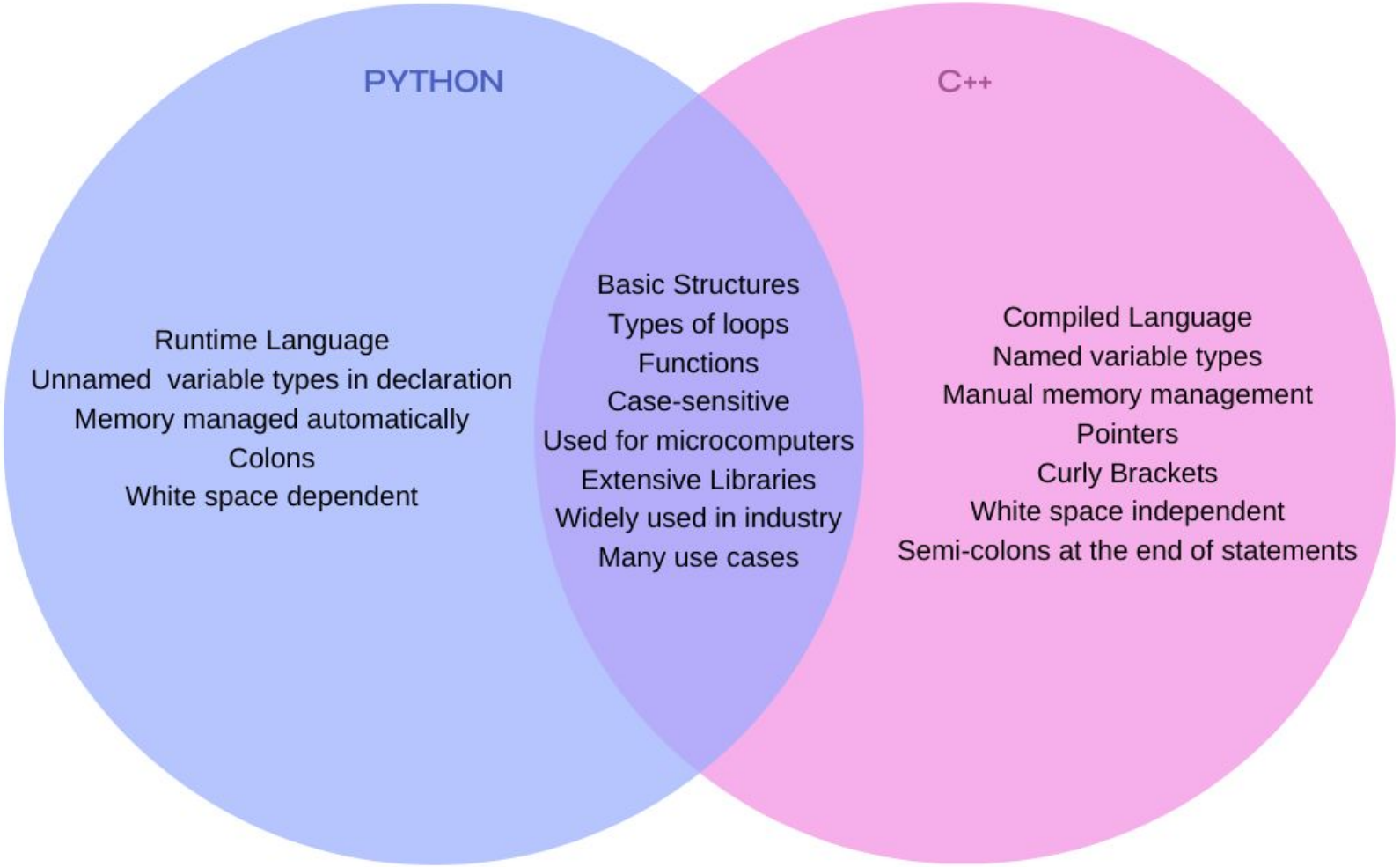
Short introduction.

More on Week 7 - Programming and Git/GitHub

Python and C++

A lot of ENGR 102 applies to C++.

There are a few new ideas but most are closely related



Syntax

The syntax of C++ is similar to the syntax of Python, with some key differences:

- All lines of code **must** end with a semicolon
- Whitespace like indents and new lines do not matter to C++
  - You should still make your code easy to read though
- Rather than using indents for loops, if-else, functions, etc., you use {curly braces}
- When creating variables, you must specify the type of variable
  - string, char, int, float
  - This is called static typing (as opposed to Python's dynamic typing)

Variable Declaration

Python

```python
num = 100
dec1 = 100.5
dec2 = 100.5
letter = 'a' # “a” also acceptable
word = 'text' # “text” also acceptable
arr = [0, 1, 2]
```

C++

```cpp
int num1 = 100;
float dec2 = 100.5; // less precise
double dec1 = 100.5; // more precise
char letter = 'a'; // must be ‘a’
// requires ‘#include <string>’; way better than c-string
String letters = "text"; // must be “text”
int arr[3] = {0, 1, 2};

// alt version
int num2; // define
num2 = 100; // initialize, Required to be inside a function
```

Auto assigns type

Must assign type

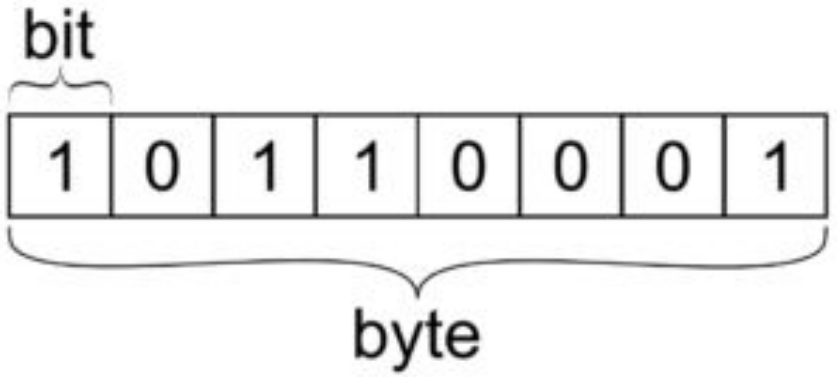
Variable Data Types

	C++ Keyword(s)	Size(s)	Description

Integers	`int`
	`short`
	`long`
	`long long`
		4 bytes
		2 bytes
		4 bytes
		8 bytes
			Holds signed numbers
			If you put `unsigned` before the keyword, will only hold numbers >= 0

Decimals	`float` `double`	4 bytes 8 bytes	float has 7 decimal digits of precision double has 15 decimal digits of precision
Boolean	`bool`	1 bit	Holds either true or false
Character	`char`	1 byte	Holds a single character
String	`std::string`	No set limit	Holds an array of characters (like a word!)

Bits / Bytes



A bit represents a single binary value

- Only two possible values
- 0 or 1, on or off, true or false

A byte is a group of 8 bits

- 2^8 (256) possible values
- Originally arose from 8 bits being used to encode a single character (the char variable type still uses this!)

It is rare to work on memory at the bit-level, most of the time you are at least working at the byte-level

For Loops

Python

```python
for i in range(0,3,1): # Start, End, Step
  print('Loop', i)
```

C++

```cpp
for(int i=0, i<3, i++){ // Start, End, Step
  std::cout << (“Loop ” + i) << std::endl;
}
```

Required to declare looping variable type

output:

Loop 0

Loop 1

Loop 2

If Else Statements

Python

```python
if grade >= 90:
    cont_dgre_pln = True
    dunk_on_friends()
elif grade >= 70:
    cont_dgre_pln = True
else:
    cont_dgre_pln = False
    cry()
```

C++

```cpp
if (grade >= 90) {
    cont_dgre_pln = true;
    dunk_on_friends();
}
else if (grade >= 70) {
    cont_dgre_pln = true;
}
else {
    cont_dgre_pln = false;
    cry();
}
```

Curly brackets {} required

Classes

Classes are an essential part of higher-level programming. They're kind of like creating your own variable type. You can:

- Choose what data your class will store
- Create functions that instances of your class can perform
  - These are typically called **"methods"**

Instances of a class are called **"objects"**. You create objects in a very similar way to how you create a variable.

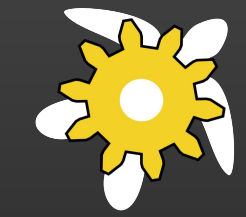
Using Classes

To create an instance of a class (an object):

- Declaration:
  - `MyClass myObj;  // create an object of MyClass called myObj`
- Initialization:
  - To initialize an object, we use the special method called the **constructor**
  - To call the constructor, we call it like a normal function that has the class's name
  - `myObj = MyClass(arguments); // data in the object will be set according to the arguments you enter`
  - To find out what arguments to use, **you should consult the documentation!**
- Using methods
  - `myObj.methodName(arguments);`
  - Again, to find out about the methods a class has, **you should consult the documentation!**

Best Practices

- Plan **before** you code
- Develop code iteratively
- Comments should only explain that which is not obvious
  - Abstracted description of what a class/function does too
  - **Do not** copy the comment style in our example code. It is terrible for a real piece of code.
- Progressively indent loops and anything in curly brackets
- Add line breaks between unrelated thoughts
- Descriptive variable names (a, b, c are not good names)



TEXAS A&M UNIVERSITY

ROBOTICS TEAM & LEADERSHIP EXPERIENCE

Signal Processing

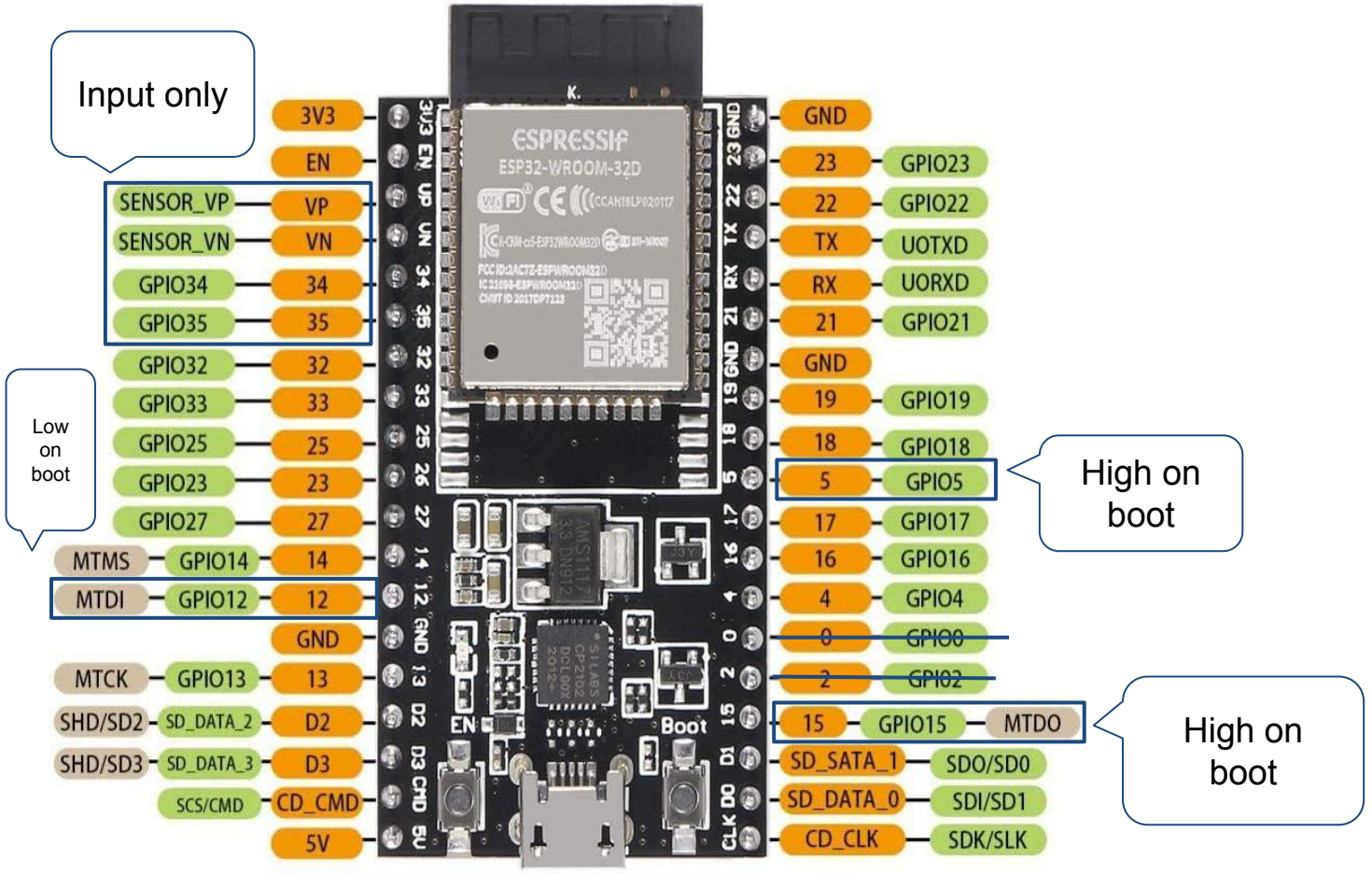
GPIO Pins

At the most basic level, the microcontroller either:

- Outputs a voltage to a pin
  - Often called a “**write**”

- Measures an input voltage from a pin
  - Often called a “**read**”

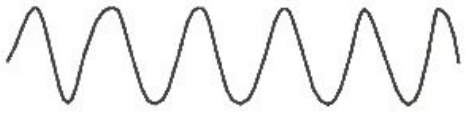
These voltages are called **signals**



Types of signal

Analog

- Continuous
- How the real world works
- Infinitely many data points
- Signal noise may corrupt data
- For example, could spin a motor at variable speeds



Analog Signal

Digital

- Discrete, most commonly either 0 or 1 (called a bit)
- How computers work
- Higher sample rate better replicates continuous data
- For example, could turn an LED either off or on

Digital Signal

Communication

- Serial Communication
  - A series of bits sent over a single wire
- SPI
  - Fast
  - Requires four wires {SCLK, MOSI, MISO, SS (CS)}
- I^2C
  - Better power consumption
  - Better for large number of peripherals
  - Requires two wires {SDA, SCL}

Basic Code Example

```ino
#include <Arduino.h> // Include the Arduino library functions

// put function declarations here:
#define LED 12 // Pin number of the LED

void setup()
{
  // put your setup code here, to run once:
  Serial.begin(115200);             // Start the Serial Monitor with the specified baud rate (monitor_speed)
  Serial.println("Setup complete"); // Troubleshooting message to indicate that the setup is complete
  pinMode(LED, OUTPUT);             // Set the LED pin as an output
}

void loop()
{
  // put your main code here, to run repeatedly:
  digitalWrite(LED, HIGH);   // Turn the LED on by setting the pin voltage output to HIGH
  Serial.println("LED on");  // Troubleshooting message to indicate that the LED should be on
  delay(1000);               // Wait for 1 second (1000 milliseconds)
  digitalWrite(LED, LOW);    // Turn the LED off by setting the pin voltage output to LOW
  Serial.println("LED off"); // Troubleshooting message to indicate that the LED should be off
  delay(1000);               // Wait for 1 second (1000 milliseconds)
}
```

Controlling electronics

Most basic tools/functions for using electronics:

- `pinMode(pin, mode)`
  - Sets a pin to either OUTPUT mode or INPUT mode
    - (or INPUT_PULLUP, dw about this for now)
  - Remember to do this in `setup()` for any pins you're using

- `digitalWrite(pin, value)`
  - "Writes" either a 0 or a 1 (LOW or HIGH) voltage to the specified pin

- `digitalRead(pin)`
  - "Reads" either a 0 or a 1 (LOW or HIGH) voltage from the pin

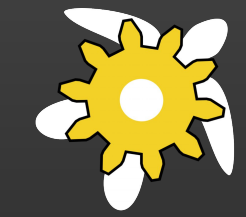
Other useful functions

Some other useful functions

- `Serial.begin(baud_rate)`
  - Starts the serial monitor at the specified baud rate (determines how many bits per second are sent)
  - Remember to do this in `setup()` if you plan on printing any debugging messages, and update platformio.ini with:
    - `monitor_speed = baud_rate`

- `Serial.println(printable)`
  - Prints out a message or value onto the serial monitor
  - Must have used `Serial.begin(baud_rate)` before using this

- `analogRead(pin)` and `analogWrite(pin, value)`
  - Analog versions of digitalRead and digitalWrite, probably won't have to use these
  - Only work on certain pins, check the ESP32 pinout before using



TEXAS A&M UNIVERSITY

ROBOTICS TEAM & LEADERSHIP EXPERIENCE

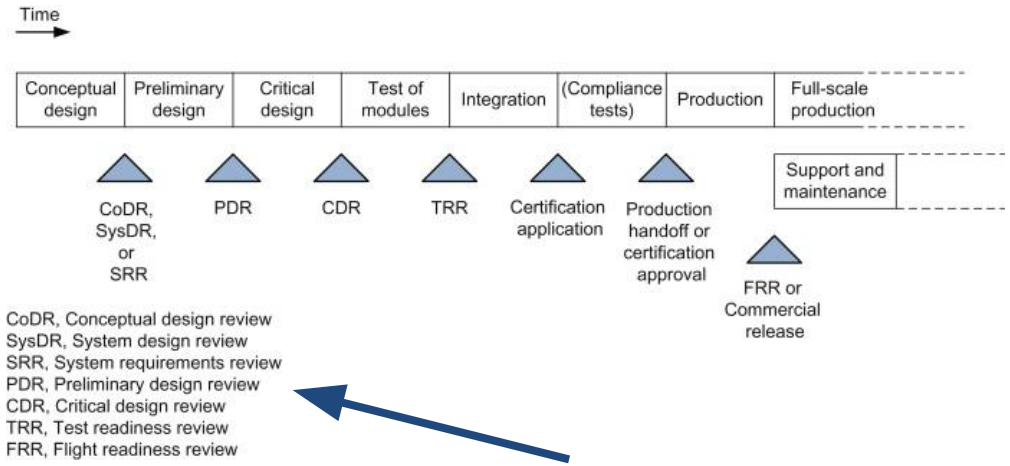
Design Review

What is a Design Review

It is an opportunity to get a new set of eyes on your solution and improve the design

Design Reviews often target

- Scope Creep
- Team alignment
- Potential issues and prevention
- Often happens in industry!
    - good to experience



Note: Aerospace is particularly structured with design reviews. SRR, PDR, CDR, and FRR are extremely common

Important things to consider

- Manufacturing
  - Can parts be easily 3D-Printed?
- Sensors
  - How are they integrated and what is the input/output?
- CAD
  - Are the COTS item CAD available online
- Initial ideas:
  - How did you approach the concept?

We Lied. Well kinda…

You'll actually be completing the project in teams of **3**. You do get to pick your own teammates. Just make sure you are under the **same Hatchling organization**.

Team formation will be sent in your orgs communication platform

Why?

- Advanced projects are team based. Developing soft skills is just as important as technical skills
- Spreads out the workload
- Collaboration inspires innovation
- Financially responsible :)

Collaboration Recommendations

Communication: Create a private discord message chat or text group chat

CAD: Create a shared drive among the team. Members can access SolidWorks files via GOOGLE Drive App without downloading/uploading files. (Cannot download on VOAL)

Code: Github (Discussed in Week 7 - Programming and Git/GitHub)

TEXAS A&M UNIVERSITY
ROBOTICS TEAM & LEADERSHIP EXPERIENCE

Breakout Groups!

We will look over y'all's designs to give some feedback, nothing crazy but just to get you guys on the road!

Action Plan

Split the room into five!

One section will verify software installs
Others will be doing the design review

Everyone will:

- Have their individual designs reviewed by a director
  - We will spend roughly five minutes on each design. Feel free to ask for more insight after everyone has gone.
- Verify GitHub account and PlatformIO VS Code setup
- Begin team formation

Next Milestone

Milestone: Assembly Review

Date: Week 7: Programming and Git/GitHub (2 weeks from today)

Expectation: Have a detailed sketch and begin CAD of the drive system. Decide on the electronics you will use.

Exceed Expectation: Have a CAD assembly of a drive system. Have a finished electronics wiring diagram.

Impact: We will review design viability and suggest improvements. Potential to prototype your mechanism.

TEXAS A&M UNIVERSITY
ROBOTICS TEAM & LEADERSHIP EXPERIENCE

SolidWorks Assembly

Next Week

Don’t forget to fill out the team formation form!

“Form Follows Function”

Louis Sullivan 1896

Hatchling