TEXAS A&M UNIVERSITY

ROBOTICS TEAM & LEADERSHIP EXPERIENCE

Project Design Tips

Hatchling

Disclaimer

This is a brief summary of information and common question answers you may need for the Hatchling project

Please see the dedicated week slides or search online for more in depth information

Always double check your work. It can save you hours of debugging

TEXAS A&M UNIVERSITY
ROBOTICS TEAM & LEADERSHIP EXPERIENCE

Reference Documentations

Navigating documentation is **VITAL** to engineering

Reference Documentations

- ESP32-WROOM-32D: https://randomnerdtutorials.com/getting-started-with-esp32/
- Motion:
  - L298N: https://lastminuteengineers.com/l298n-dc-stepper-driver-arduino-tutorial/
  - tt-motors: https://www.adafruit.com/product/3777
  - SG90: https://protosupplies.com/product/servo-motor-micro-sg90/
- Sensors:
  - HC-SR04: https://lastminuteengineers.com/arduino-sr04-ultrasonic-sensor-tutorial/
  - AS5600: https://www.instructables.com/AS5600-Magnetic-Angle-Encoder/
  - MPU6050: https://lastminuteengineers.com/mpu6050-accel-gyro-arduino-tutorial/

TEXAS A&M UNIVERSITY
ROBOTICS TEAM & LEADERSHIP EXPERIENCE

CAD

Design

- 2 - 5 mm thick parts depending on expected load
    - 5 mm will not break for Hatchling applications
- Fillet at sharp corners. Chamfers to reduce overhangs
- No parts larger than the build volume (256 mm^3)
    - Aim for 250 mm^3 or less. Slicer does NOT like 256 mm^3
- Start with a 0.2 (0.4 total) mm clearance on parts
    - This is a transition fit
- Use McMaster-Carr and other websites for COTS CAD models
    - Lab stock include various lengths of M2, M2.5, M3, and M4 screws

Slicer Settings

Printers: X1-Carbon 0.4 mm / P1S 0.4 mm / P1S 0.6 mm

Plate: Textured PEI

Material: Generic PLA / BAMBU PLA / BAMBU PETG HF

Infil: 15-25% (15% for most parts), Gyroid Pattern

Supports: Tree(auto) if necessary

TEXAS A&M UNIVERSITY
ROBOTICS TEAM & LEADERSHIP EXPERIENCE

Electronics

Motion Devices

tt-motor (DC Motor)

- Continuous rotation
- Low torque / High speed
- Requires a L298N Motor Driver to stop/start/change directions

SG90 (Micro Servo

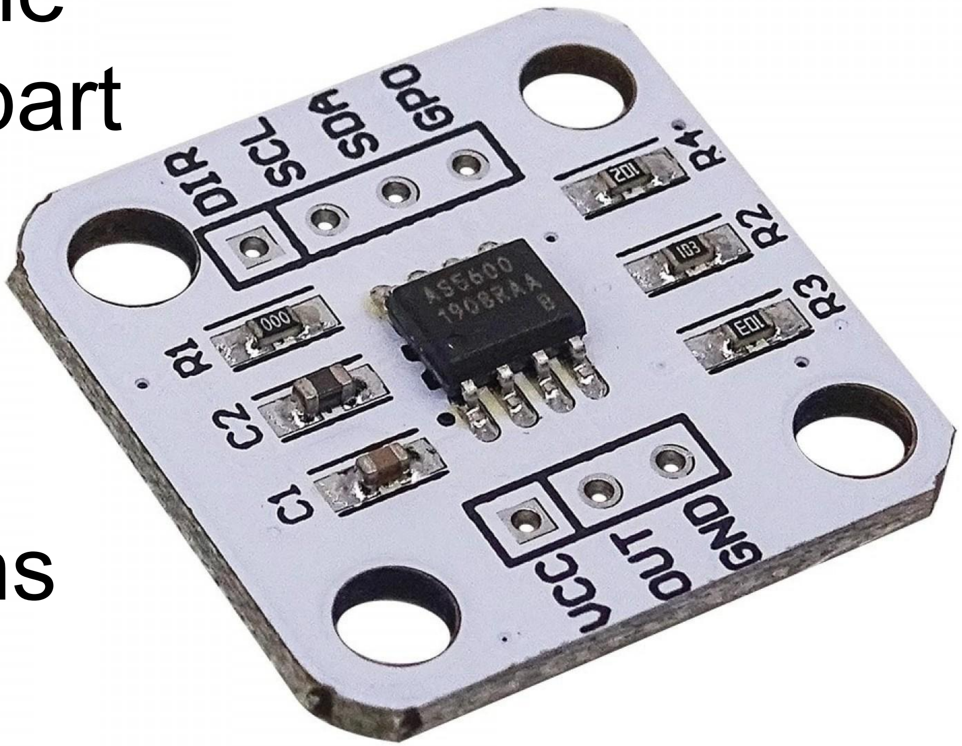
- 180° limited rotation
  - Goes to programmed angle
- High torque / Low speed
- Controlled directly from ESP32 through PWM

HC-SR04 Ultrasonic Distance Sensor

- Detects object distance by timing the echo of an ultrasound wave burst
- Ultrasound waves are sounds with frequencies above human hearing range
- Range of 4 cm to 4 m
- Accuracy of 3 mm

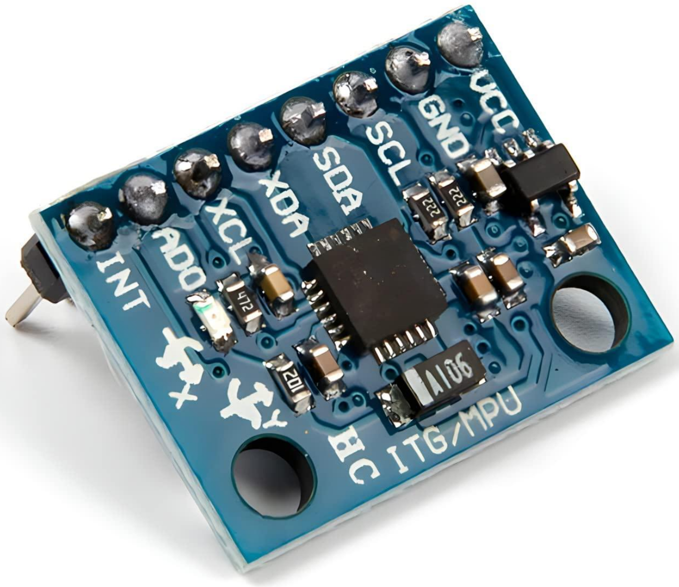
AS5600 Magnetic Encoder

- Uses a magnet to determine angle without making contact with the part (digital potentiometer)
- Returns absolute angle measurements from 0° to 360°
- Programmable start stop positions
- Uses I^{2}C interface
- Variations up to 2°



MPU6050

- Gyroscope, Accelerometer, and Digital Motion Processor
- Gyroscope can read up to ±2000° /sec
- Accelerometer can read up to ±16g
- Programmable filters
- Uses I^2C interface
- Programmable range



General

- Do NOT power ANY board by the GPIO and USB Port at the same time
- Do NOT supply higher voltage than recommended
- Electronics may get warm, unpower if abnormally hot
- A stripped solid core wire acts as a male jumper end

Hardware

See documentation on proper wiring

- GND: Grounds should all be connected (Common)

- All 5 V power should come from the L298N 5 V pin
    - 5 V pin on ESP32 should be used to power the ESP32

- 7.2 V Battery should go into the L298N 12 V and GND pin

TEXAS A&M UNIVERSITY

ROBOTICS TEAM & LEADERSHIP EXPERIENCE

Programming

GitHub Commands

git clone <link> : Creates a local workspace from remote repository

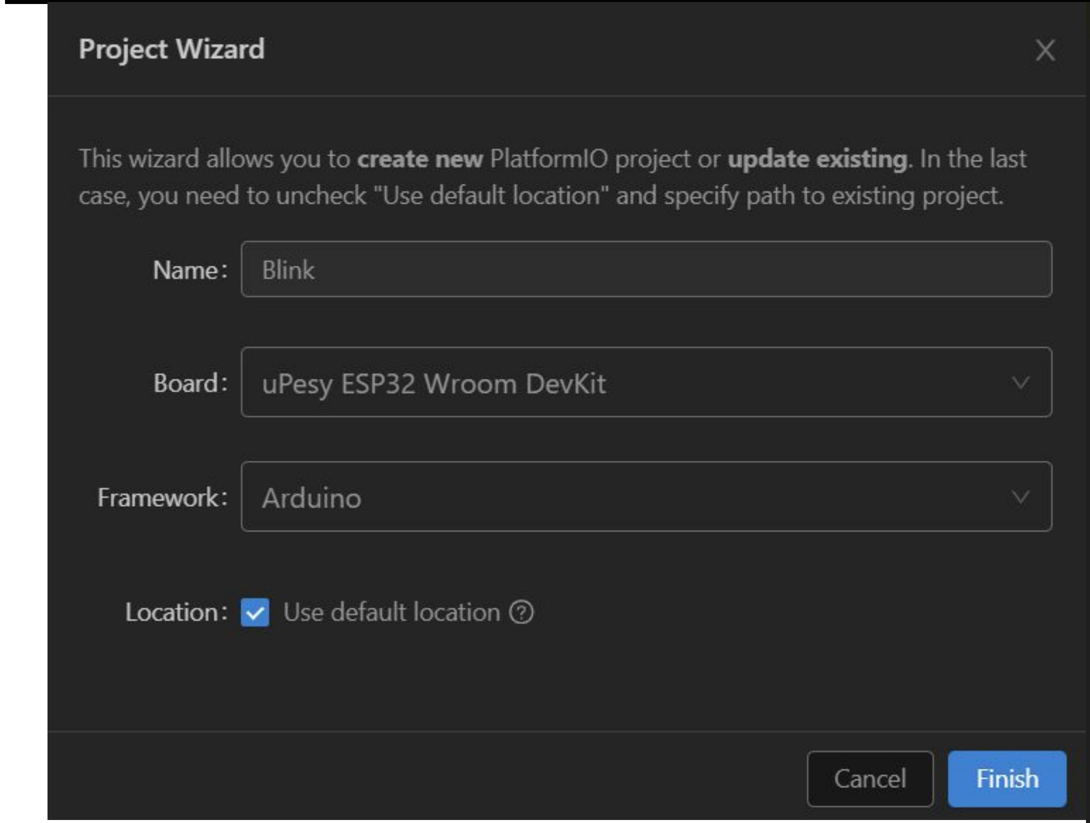
git add . : Stages all files

git commit -m “<message>” : Staged files to local repository

git push : Local repository to remote repository

git pull: Fetches and merges remote repository to local workspace

PlatformIO and Code Layout



Project Wizard

This wizard allows you to **create new** PlatformIO project or **update existing**. In the last case, you need to uncheck "Use default location" and specify path to existing project.

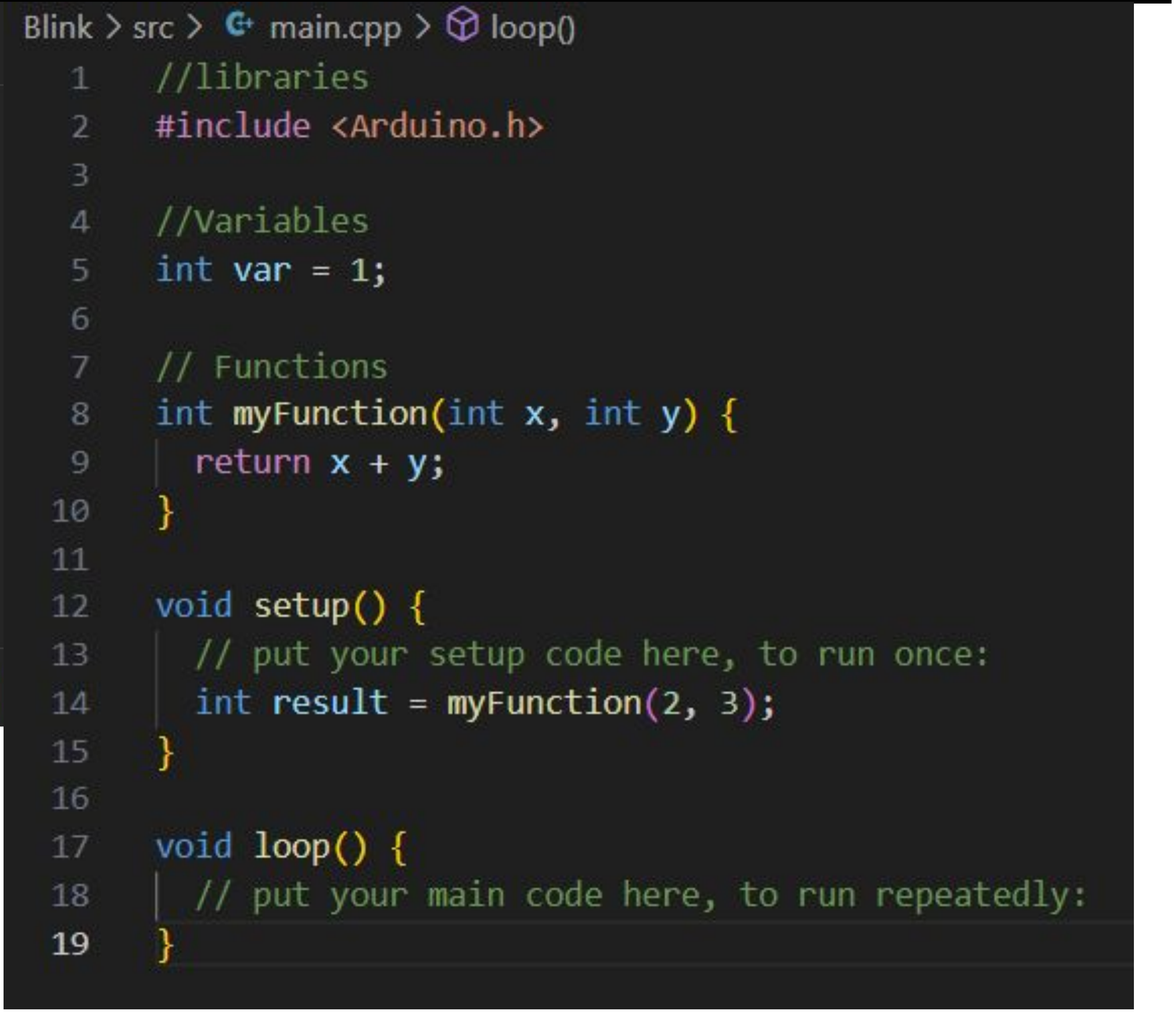
Name: Blink

Board: uPesy ESP32 Wroom DevKit

Framework: Arduino

Location: ☑ Use default location

Build then Upload code



Blink > src > main.cpp > loop()

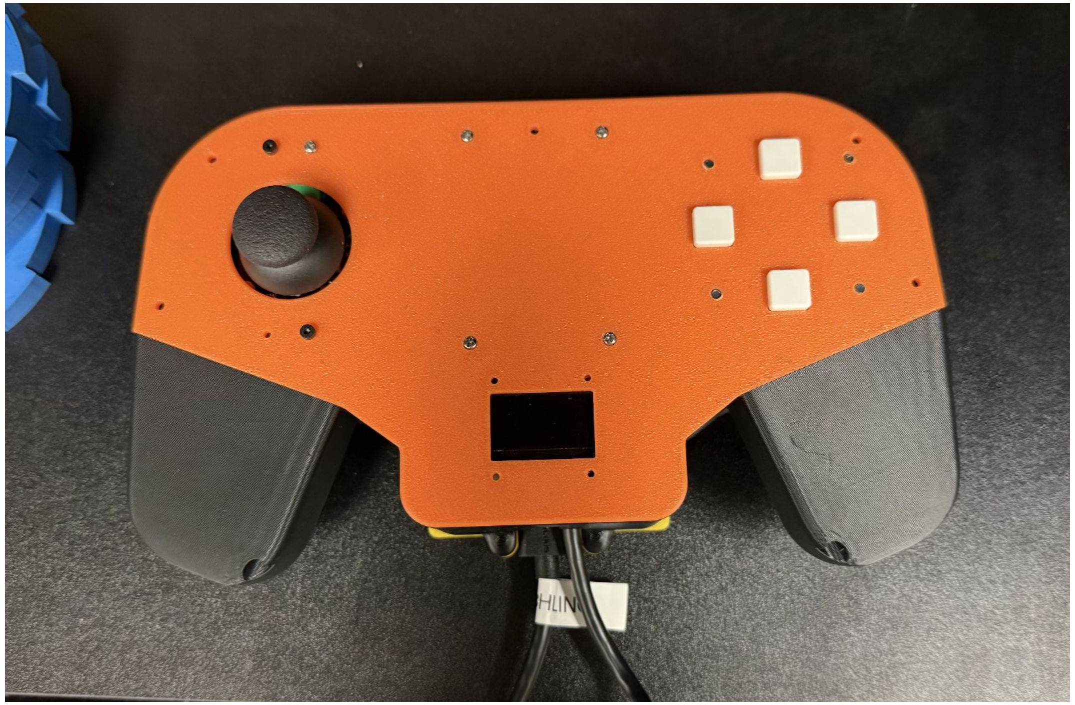
```
//libraries
#include <Arduino.h>

//Variables
int var = 1;

// Functions
int myFunction(int x, int y) {
  return x + y;
}

void setup() {
  // put your setup code here, to run once:
  int result = myFunction(2, 3);
}

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

TEXAS A&M UNIVERSITY
ROBOTICS TEAM & LEADERSHIP EXPERIENCE

HURC Controller

The Hatchling Universal Remote Controller

BEHOLD



One joystick, two shoulder buttons, four face buttons
(very cool)

Importing the library

To receive inputs from the controller, you'll have to import our very own TurtleReceiver library.

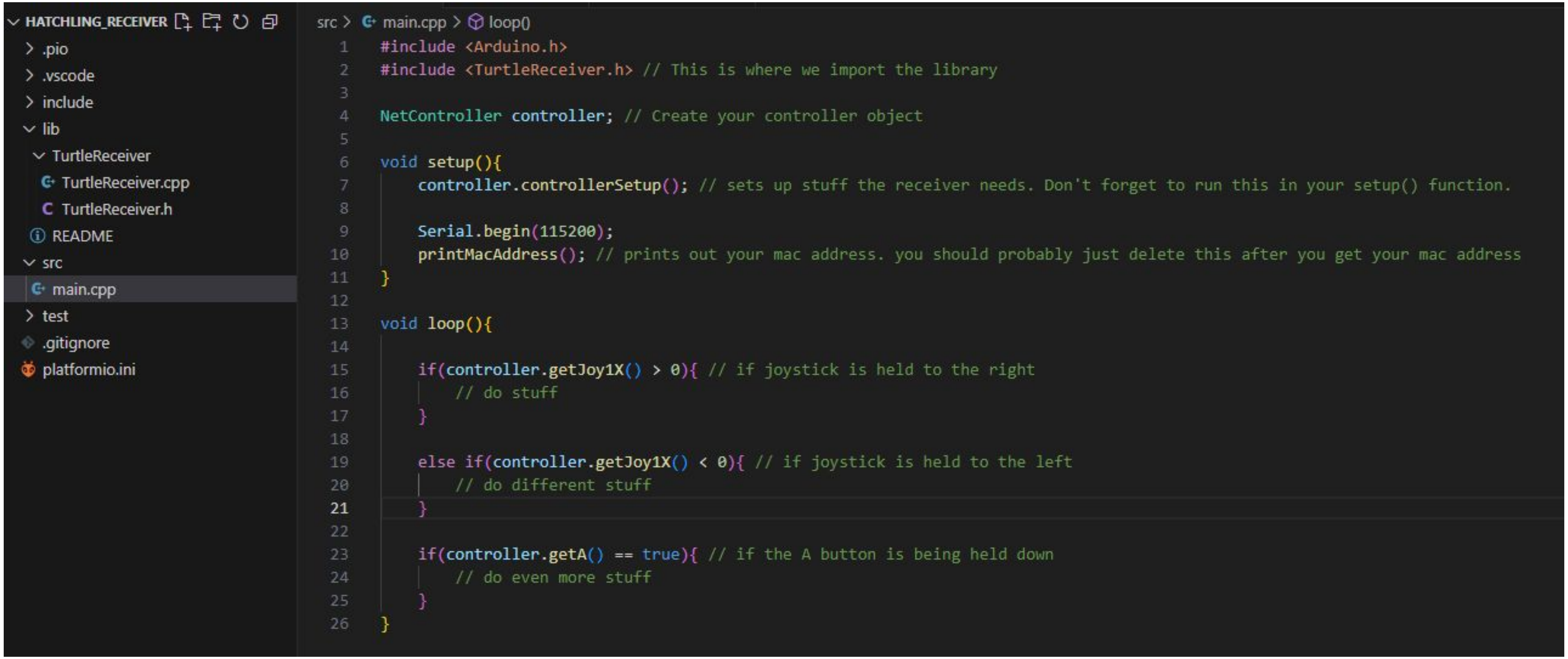
To do so:

1. Download TurtleReceiver.zip (also in the google drive)
2. Extract the ***TurtleReceiver*** folder and place it in the "***lib"*** folder of your PlatformIO Project
   a. If PlatformIO cannot find the library, move **TurtleReceiver** to "***src***"
3. Put `#include <TurtleReceiver.h>` at the top of your main.cpp file

Example Code

This is where the downloaded folder goes!

If the above arrow gives errors, it goes here



```cpp
#include <Arduino.h>
#include <TurtleReceiver.h> // This is where we import the library

NetController controller; // Create your controller object

void setup(){
    controller.controllerSetup(); // sets up stuff the receiver needs. Don't forget to run this in your setup() function.

    Serial.begin(115200);
    printMacAddress(); // prints out your mac address. you should probably just delete this after you get your mac address
}

void loop(){

    if(controller.getJoy1X() > 0){ // if joystick is held to the right
        // do stuff
    }

    else if(controller.getJoy1X() < 0){ // if joystick is held to the left
        // do different stuff
    }

    if(controller.getA() == true){ // if the A button is being held down
        // do even more stuff
    }
}
```

Getting your ESP32 MAC Address

For the controller's ESP32 to connect to your robot's ESP32, it needs your ESP32's MAC address. The library includes a function called `printMacAddress()` which you can put in your `setup()` function to have the board print out your MAC address to the serial monitor.

Don't forget to run `Serial.begin(115200)` before calling `printMacAddress()` and to update your platformio.ini file with `monitor_speed = 115200`

Basic Usage

Similar to other libraries, this library works by creating a object of a class for your code to interact with. For this library, we have the *NetController* class.

The *NetController* class will handle all the input-receiving code for you, you just need to create an object **and then make sure to run the `controllerSetup()` member function** in your `setup`() function.

From there, there are a host of ‘getter’ functions to tell you the current state of the buttons and joysticks.

Documentation

For info on the rest of the joystick stuff/buttons, consult the documentation.

(the printMacAddress() function isn’t listed since it isn’t a part of the class, but I promise it’s in there)

If you have any questions about using the controller/how it works (or if you find a bug 😭), ping or DM a TURTLE Hatchling Director on discord.

“The more you learn, the more you earn”

Warren Buffett

Hatchling