3DoT Board

Release 3.1

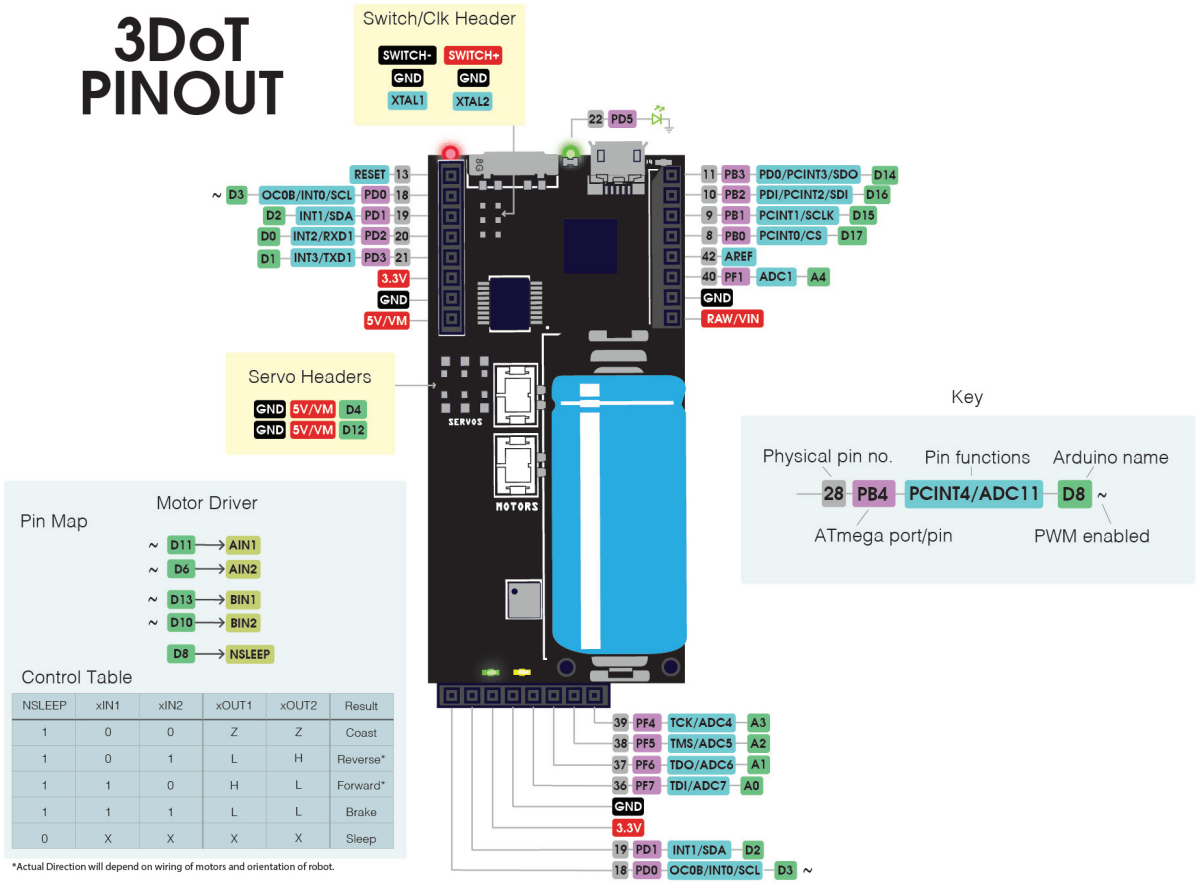
May 03, 2021

Contents:

CHAPTER 1

The 3DoT Board

1.1 Pinout



1.2 Battery

When you first get your board, the battery will be at an ~80% charge, or about 3.7V. Plug in a usb cord connected to a computer or USB wall charger to charge the battery. The 3DoT Board will automatically stop charging at 4.2V, the maximum safe charge voltage.

Attention: Do not let the battery discharge below 3.2V, as this could cause permanent damage to the battery's chemistry and significantly reduce battery life.

1.3 ON - PRG - RUN Switch

Next to the USB port, the 3-position switch on the 3DoT board is designed to allow you to set the board to programming mode, take your time programming the board, and then switch the board to ON when you are ready to run the code.

Note: In order to go from ON to programming the board, you must first turn the board OFF, then switch back to PRG.

Fig. 1: ON - PRG -OFF Switch

1.4 Motors

The 3DoT motor driver supplies 0-5V to the on-board motor connectors. Any 3-6V motors will work, as long as they are not too large (no-load current above ~200mA).

We highly recommend micro-metal gear motors.

1.5 Servos

Servos on the 3DoT are powered from the same VM net as the motor drivers. Again, any 3-6V servos should work, but be careful with amount of current that can be drawn from the on-board battery.

We recommend any kind of "micro-servo".

1.6 External Battery Connector

Depending on the motors and servos used, the amount of torque required from them, current drawn by shields etc. you may hit the 3DoT battery's 2A limit. You can alleviate this problem by desoldering the marked solder bridge on the bottom of the board and connecting an external battery or other power supply (4V - 18V) to the external battery connector.This will allow the internal electronics of the board to run from the on-board battery, and supply your battery's power directly to the "VM" net, which is accessible from the top header, and powers the motor driver.

1.7 Shields

Shields add functionality to your 3DoT board, such as extra motor connectors or sensors, and are available to purchase on the Humans for Robots store, or your can make your own.

To make your own, try free PCB CAD software such as KiCad or EAGLE and use the 3DoT PCB CAD files below as a reference.

1.8 Schematis and PCB CAD Files

Schematics

Altium PCB Files

PCB Files to import to other CAD (Eagle, KiCad etc..)

CHAPTER 2

Getting Started

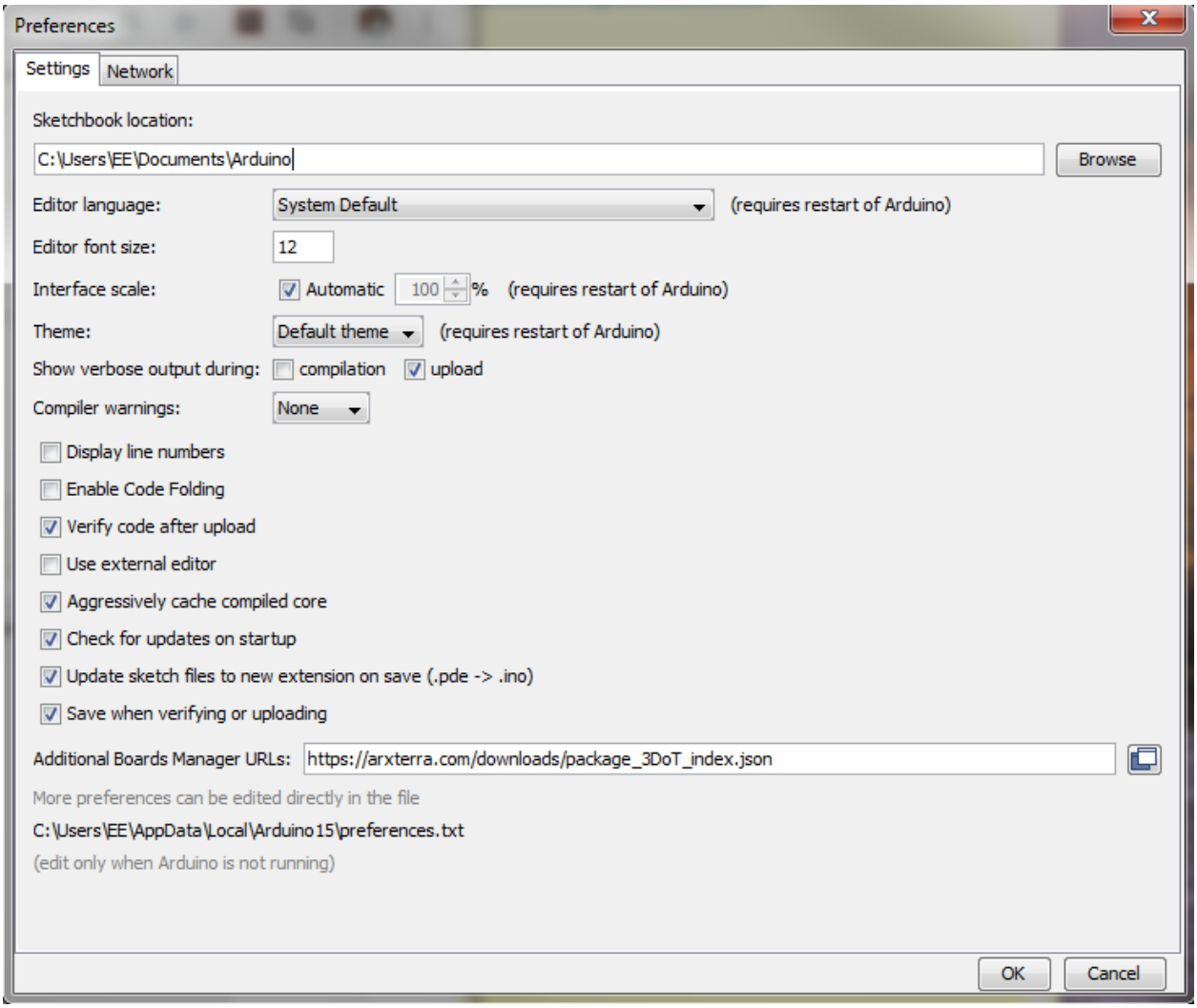
The Arduino IDE is the most popular beginner method for programming compatible microcontrollers.

Download the Arduino IDE from the Arduino website:

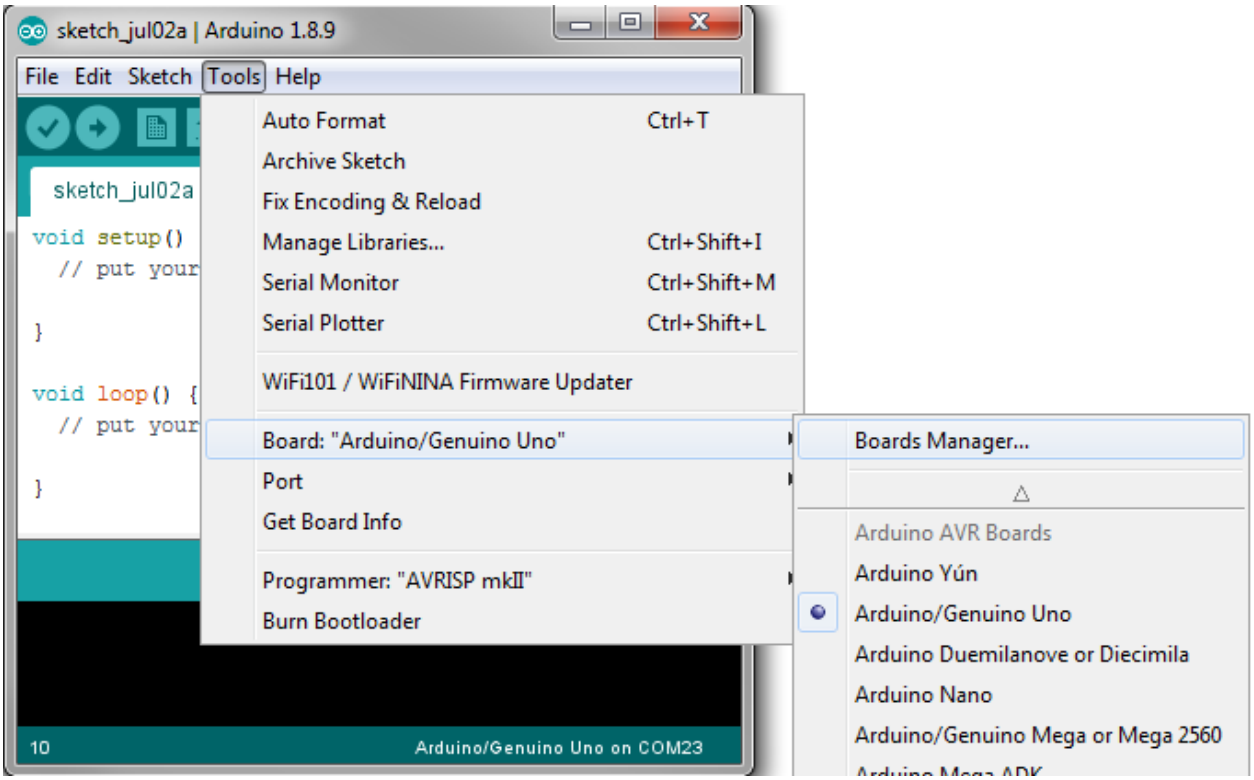
https://www.arduino.cc/en/Main/Software

2.1 Adding the 3DoT Board to Arduino

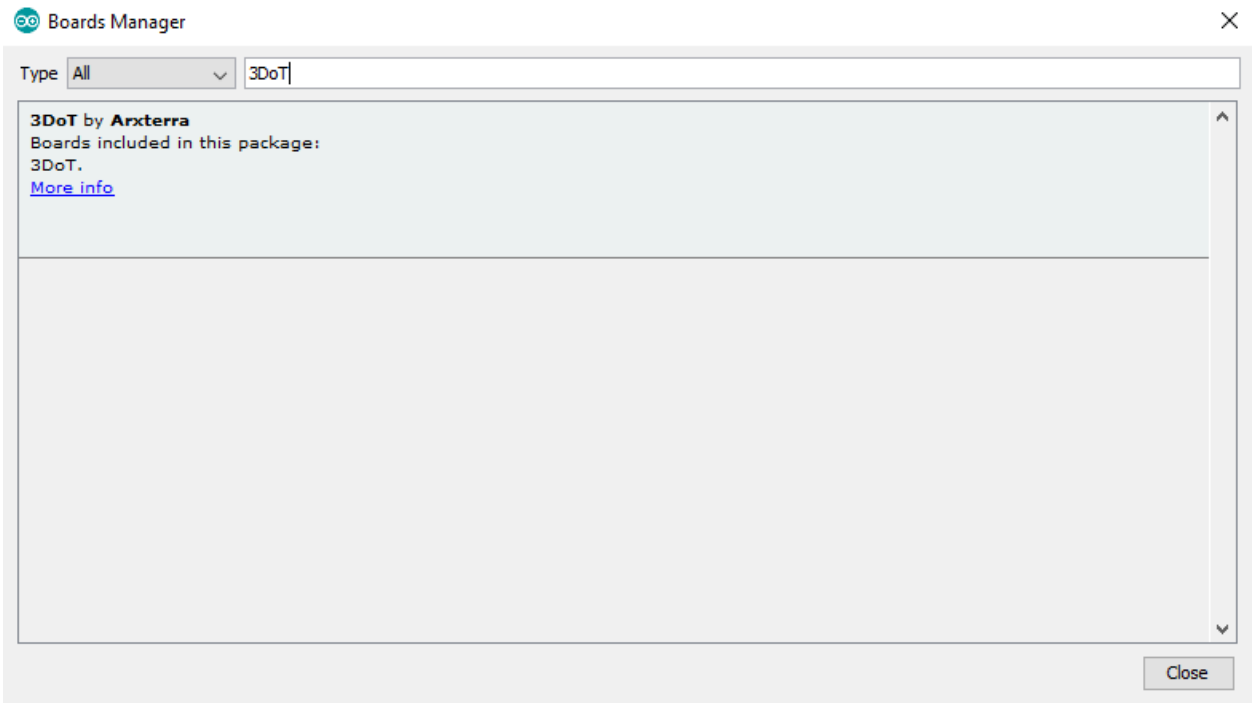
1. Navigate to File / Preferences



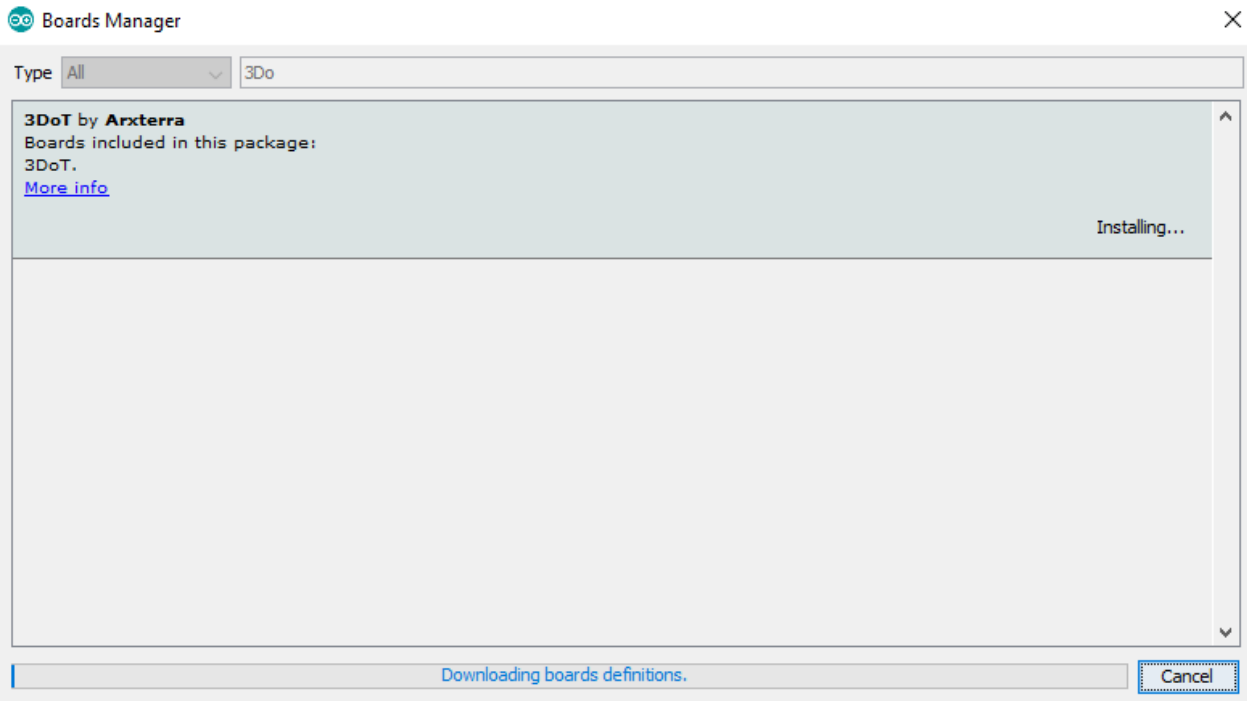
2. Paste the following link into Additional Board Manager URLs: https://arxterra.com/downloads/package_3DoT_index.json
3. Close the preferences window and navigate to Tools / Board: / Boards Manager. . .



4. Type “3DoT” in the search bar. The board Should show up.



5. Click inside of the box that contains the text “3DoT by Arxterra” and the “Install” button should appear. Click Install.

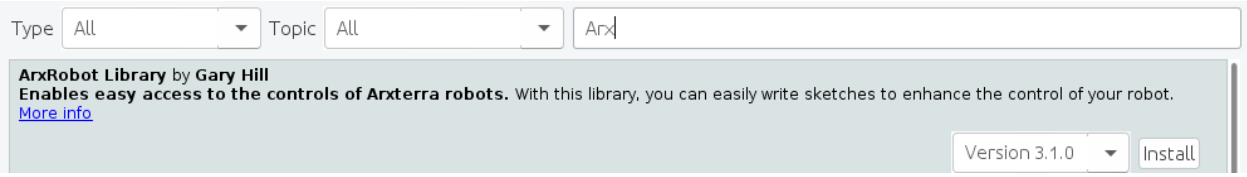


2.2 Installing the 3DoT Arduino Library

In the Arduino IDE, navigate to

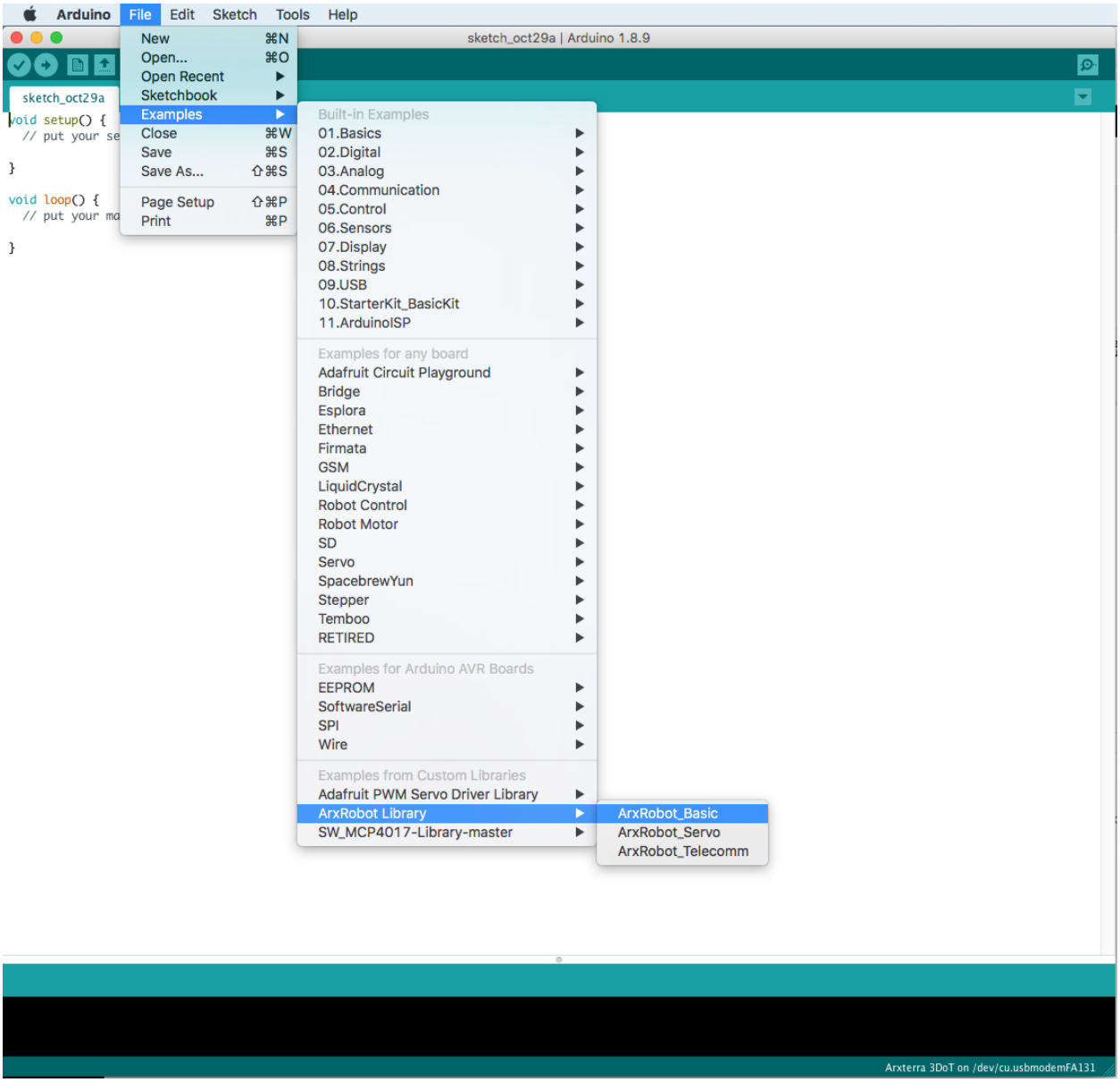
Sketch > Include Library > Manage Libraries..

Search for ArxRobot Library and click install. Done!



2.3 Uploading Your First Sketch

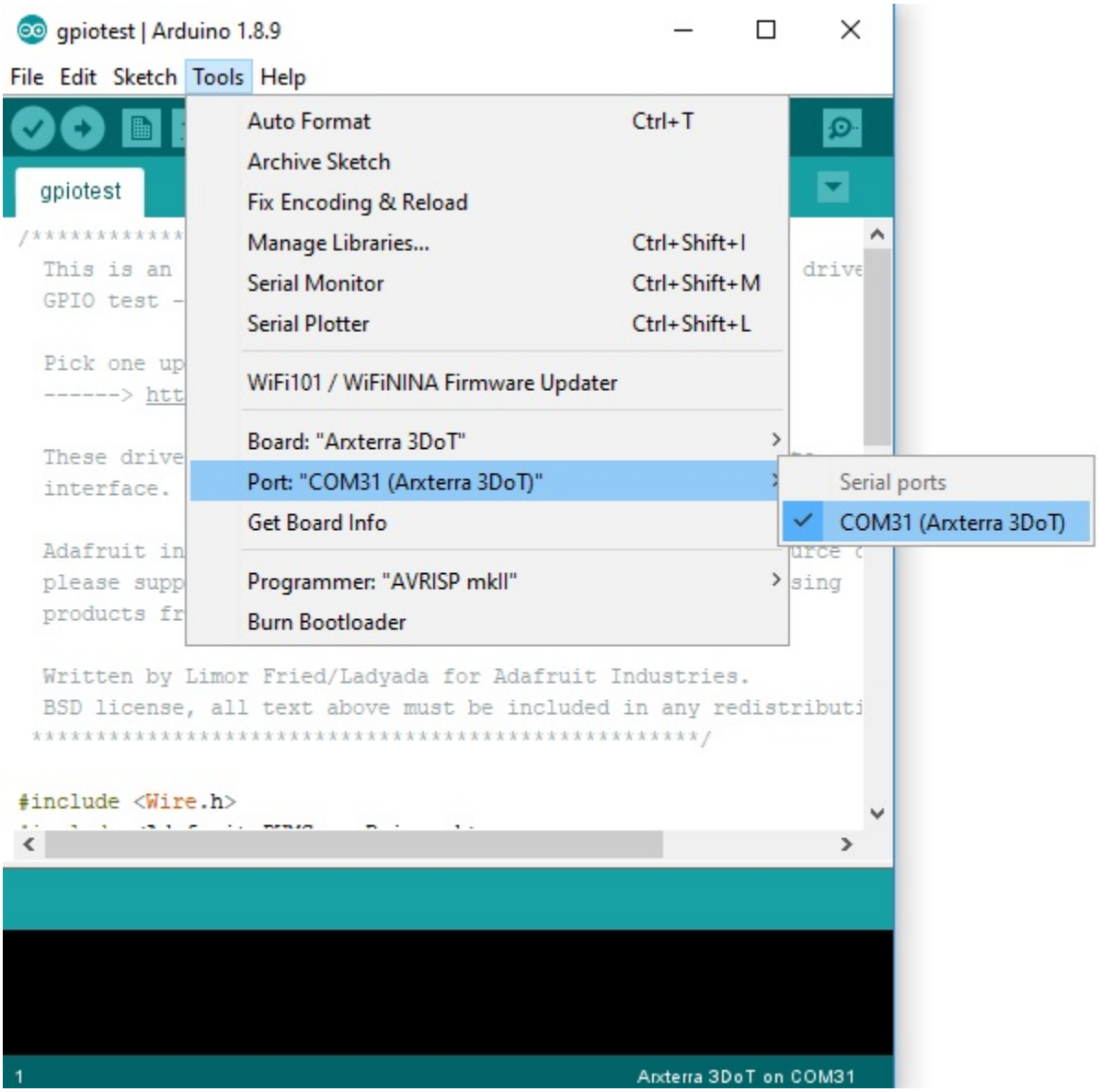
With the Library Installed, navigate to File > Examples > ArxRobot Library and click on ArxRobot_Basic



The ArxRobot_Basic example is a barebones sketch that handles all the background robot operations, including Bluetooth commands. **This means that uploading the Basic example code will make your robot ready to connect to Bluetooth and go!**

2.3.1 Select a COM port

1. Set the 3DoT to PRG mode and connect your 3DoT to your computer with a USB cable
2. Navigate to Tools > Port and select the port labeled as a 3DoT board.



Then, simply hit the Upload button in the Arduino IDE.

Note: If a sketch fails to upload, check the COM port selected again. Your computer may change the COM port number as the board is turned off/on

2.4 Next Steps

Congratulations! You uploaded your first sketch to the 3DoT Board. Click the next button to learn how to send commands to your 3DoT using the RoboPilot app.

CHAPTER 3

RoboPilot

3.1 Download

Download the RoboPilot app on the Google Play Store:

https://play.google.com/store/apps/details?id=com.HumansforRobots.RoboPilot&hl=en_US

or on the App store:

https://testflight.apple.com/join/AW41JCnI

3.2 Connecting to Your Robot

Ensure the ArxRobot_Basic sketch, or your own similar code, is uploaded to the 3DoT Board (see previous section).

Turn on the 3DoT Board and load up the app.

Tap “Scan” to look for your 3DoT. If nothing shows up, tap it again after a few seconds.

Click “Connect”.

Once connected, try out the default controls to drive your robot’s motors!

3.3 Reverse Motor Directions

Click the “Reverse Motors” button from any of the controls screens to reverse any of the motor directions, in case your robot is running backwards or rotating when you try to drive it forward.

The toggles in this menu alter only the direction data byte in the command packet sent to the robot.

3.4 Adding Custom Commands

Click the "Add Command" button from any of the controls screens and choose the type of command you wish to send.

The data packet is constructed by the app as follows for each command:

Command Type	Data size (Bytes)	Format	Example	Notes
Button	0			No data, just command sent
Toggle	1	00/01	01	Boolean
Slider	1	0-125	5F	
Text	maximum 20	ASCII	68 65 6c 6c 6f	

You only need to worry about this once you start programming the robot's responses to the commands, covered on the next page: ArxRobot Library.

3.5 Saving and Loading Commands

Once you have constructed your perfect Robot UI, you can save it by clicking "Save Commands". When ever you want, you can then load them again by clicking "Load Commands".

The D-pad and tank control screens save their commands in separate files and won't interfere with each other.

Support for saving multiple command layouts for each screen is a planned feature.

CHAPTER 4

ArxRobot-Library

Make sure you have completed the Getting Started section before diving into this section.

4.1 Overview

The ArxRobot library allows the 3DoT board to communicate over bluetooth using **command** and **telemetry** *Data Packets* with checksum.

Commands are, as the name suggests, instructions for the robot or app to do something. **Telemetry** is simply data, such as the battery voltage that is communicated.

The library contains a list of *Built-In Commands* and telemetry labels, listed in Configure.h, such as the MOVE command, which is command number 01.

The function of these commands are defined in the library. For example, when the MOVE command is received, motors connected to the 3DoT are driven at the given speed and direction.

4.2 Example Sketches

In the Arduino IDE, navigate to *File* > *Examples* > *ArxRobot_Library* to open the included example sketches.

ArxRobot_Basic contains all the code needed to control a robot using the RoboPilot app.

ArxRobot_CustomCommands implements the addCustomCommand() and replaceBuiltInCommand() methods explained below to make user commands work with the RoboPilot app.

ArxRobot_ledPWM shows a simple custom command example, using a slider to adjust the brightness of an LED

4.3 Writing Custom Commands

To get started making a custom command, write a function in your arduino sketch with the following template:

```
void myCommand (uint8_t cmd, uint8_t param[], uint8_t n)
{
// Your code here
}
```

You can change myCommand to the name of your choosing, e.g. blinkLED, but the type must always be **void** and parameters must follow the **(uint8_t cmd, uint8_t param[], uint8_t n)** format.

You can use these parameters in your function.

void **myCommand**(uint8_t *cmd*, uint8_t *param*[], uint8_t *n*)

Parameters

- **cmd** – The command ID of your custom command
- **param[]** – Command data sent by the app
- **n** – Number of bytes of command data received in param[]

For example, in the ArxRobot_ledPWM example we adjust the brightness of an LED using the slider value sent from the RoboPilot app as follows:

```
void ledPWM (uint8_t cmd, uint8_t param[], uint8_t n)
{
    // param[0] is the first byte of command data received from the app,
    // which holds the slider value
    analogWrite(LED, param[0]);
}
```

4.4 Adding Custom Commands

Next, to add the function as a custom command that the app can call, simply call the following function in the setup portion of your code:

void `ArxRobot`**::addCustomCommand**(fptr_t *function*, uint8_t *commandID*)

Parameters

- **function** – Name of function to add to custom commands list
- **param[]** – Command ID to associate with function

For example:

```
void setup()
{
    ArxRobot.begin();

    ArxRobot.addCustomCommand(myCommand, 0);
    ArxRobot.addCustomCommand(myOtherCommand, 1);
}
```

In the setup() of this sketch, we are adding “myCommand” and “myOtherCommand” as two custom commands, with IDs 0 and 1.

If you now create a custom command in the RoboPilot app with ID 0, it will call myCommand, and a command with ID 1 will call myOtherCommand.

4.5 Built-In Commands

In its current state, the RoboPilot App only uses the Move and Ping Commands. Functionality for other commands in being ported over from the previous app.

Name	ID	params[]
MOVE	0x01	[0] = Left Direction, [1] = Left Speed, [2] = Right Direction, [3] = Right Speed
CAMERA_MOVE	0x02	[0-1] = Pan, [2-3] = Tilt
CAMERA_MOVE_HOME	0x03	
CAMERA_MOVE_RESET	0x04	
READ_EEPROM	0x06	[0-1] = Address, [2] = Number of Bytes to Read
WRITE_EEPROM	0x07	[0-1] = Address, [2] = Number of Bytes to Write, [3-(2+N) = Data
SAFE_ROVER	0x08	
SLEEP	0x0A	
WAKEUP	0x0B	
HEADLIGHT_OFF	0x0C	
HEADLIGHT_ON	0x0D	
COMM_SETUP	0x10	[0] = Mode ID
PING	0x11	

4.6 Replacing Built-In Commands

When using, for example, the D-pad or Tank controls on the app, the built-in MOVE command is called. If you wish to make the controls call your own function instead, simply write a custom command as explained in *Writing Custom Commands*, then call:

void `ArxRobot::`**`replaceBuiltInCommand`**(uint8_t *commandID*, fptr_t *function*)

Parameters

- **`commandID`** – Command ID to associate with function. Can use macros defined above.
- **`function`** – Name of function to add to custom commands list

Example:

```
void robotWalk((uint8_t cmd, uint8_t param[], uint8_t n)
{
   // Robot walking code here
}

void setup()
{
    ArxRobot.begin();

    ArxRobot.replaceBuiltInCommand(MOVE, robotWalk);
}
```

4.7 Data Packets

Click the "Show Console" Button in the RoboPilot app to see data packets sent and received.

A Data Packet is constructed as follows:

Packet ID, **Data** Length (N), **Command** ID, **Data** 0, ..., **Data** N, **Checksum**

- **Packet ID**: 0xA5 for a command, 0xCA for a telemetry message
- **Data Length**: The number of data bytes sent in packet
- **Command ID**: The command ID. E.g. 0x01 for MOVE
- **Data[n]**: Data bytes
- **Checksum**: XOR all the previous byte values to obtain checksum

4.8 Library Reference

4.8.1 Configure.h

Contains global flags, pin definitions and command/telemetry IDs

4.8.2 ArxRobot

4.8.3 Motor

4.8.4 Packet

4.8.5 Servo3DoT

4.8.6 Telecom

4.8.7 Watchdog

Index