HullPixelbot Architecture Overview

Version 1.0

Rob Miles

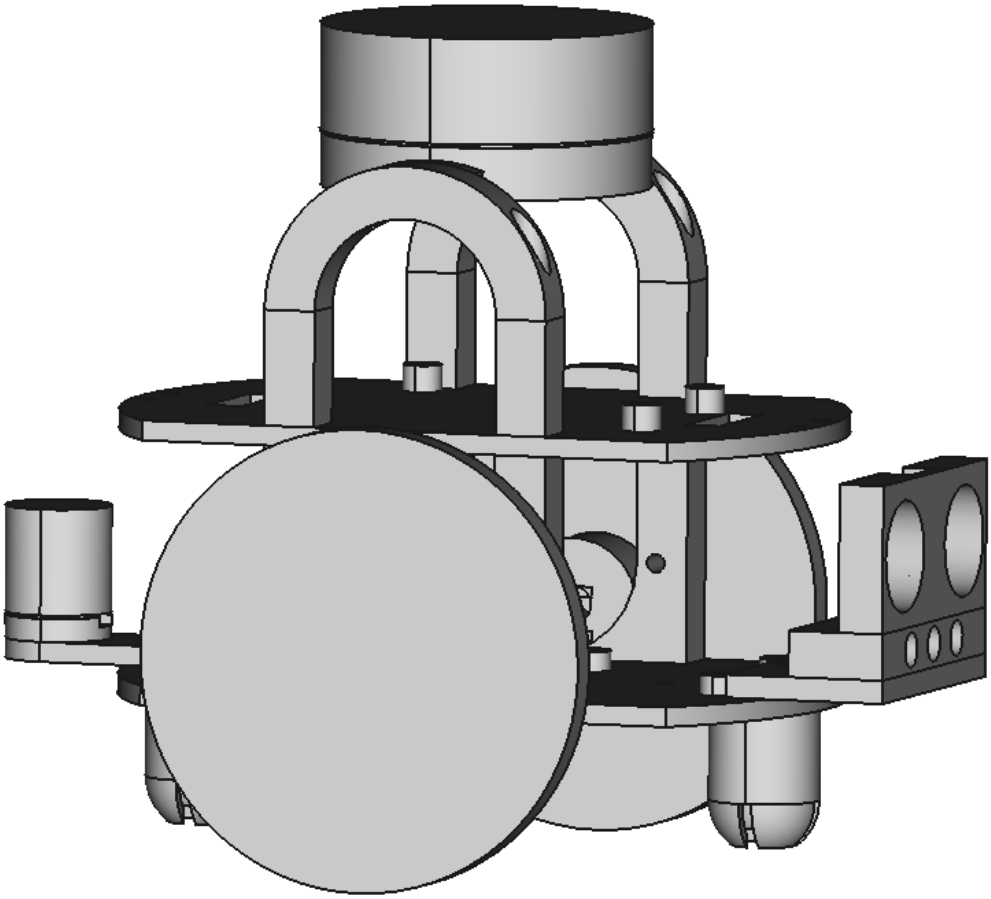
Introduction

This document describes the architecture of the HullPixelbot. It contains references to the GitHub repositories which hold the various elements of the project. The document is organised into the following sections

- Hardware designs: the files that can be used to 3D print a HullPixelbot chassis
- Robot Architecture: the processors and applications that run on the robot directly
- Azure Event Manager: a Microsoft Bot Framework chatbot that manages a number of robots that can be individually programmed
- HullPixelbot Code Editor: a Windows application that allows the editing of HullPixelbot code programs and pixel designs.

Hardware designs

This is a complete set of files which can be used to print a complete HullPixelbot chassis.

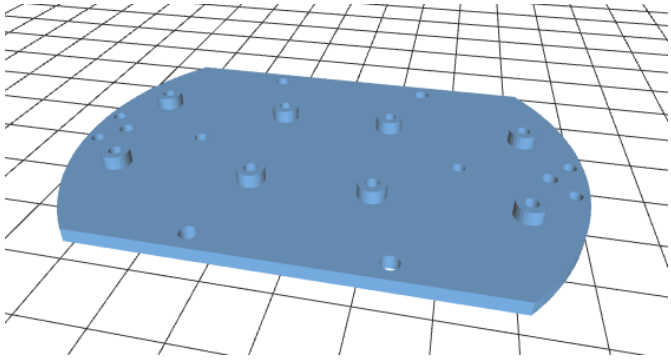


I've found that they print correctly with layer heights up to 0.2mm. It will take around 6 hours to print a complete robot. The files you need for each element are as follows:

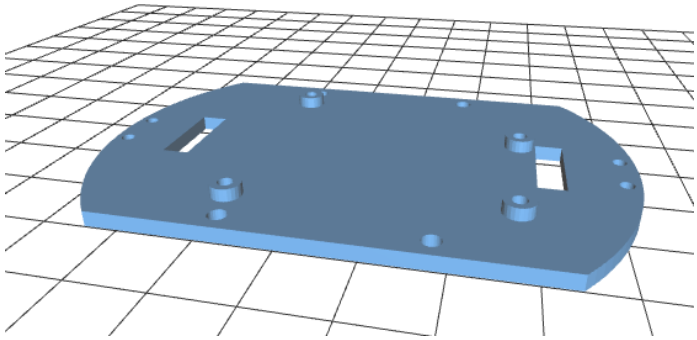
HullPixelbot Chassis:

All robots need the chassis. This holds all the elements of the robot. Other parts can be added later.

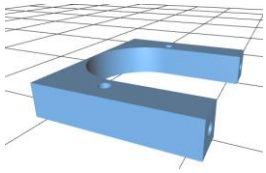
base.stl: the base that takes the stepper driver boards



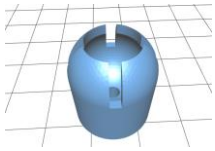
top.stl: the top that takes the Arduino



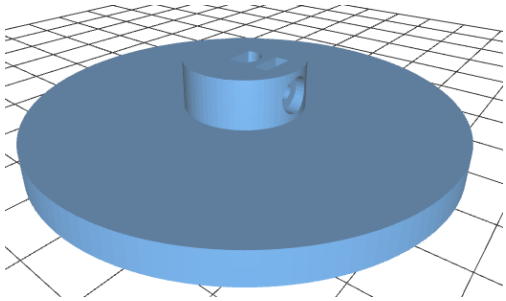
MotorTower.stl: fits between the base and the top, and provides fittings for the stepper motors. Print 2 of these



skid.stl: provides a skid that fits between the base and the ground, and stops the robot from tipping when it moves. Print 2 of these



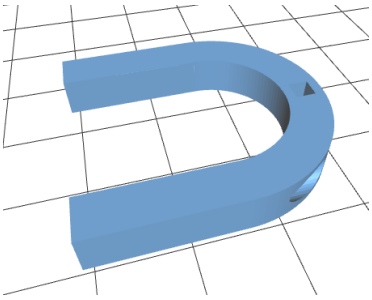
wheel.stl: the robot wheel. I'd advise printing 2 of these.



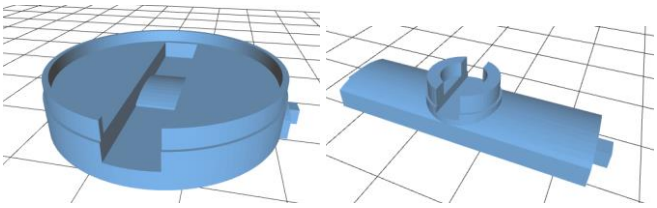
HullPixelbot Pixel:

You can add a “pixel” to top of the HullPixelbot. There are two designs; one for a single pixel and one for a ring of pixels. These fit across the top of the robot and are designed to be easy to “un-clip” without undoing any bolts.

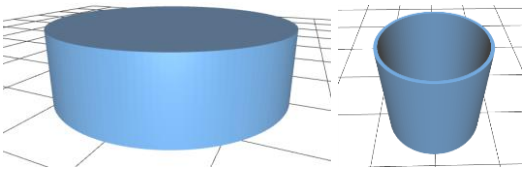
handlesupport.stl: fits on top of each motor tower and provides a fitting to insert the pixel ring support. You’ll need 2 of these.



ringcrosssupport.stl or **ledcrosssupport.stl**: fits between the two handle supports and can hold a single neopixel or a pixel ring. The square pegs on each end of the support engage with the holds in the handlesupport.



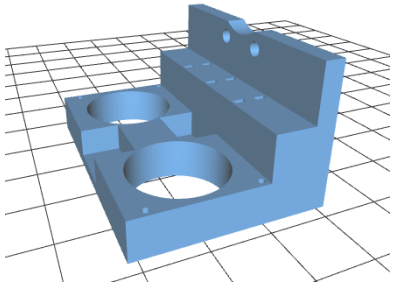
ringshade.stl or **shade.stl**: a shade that fits over the cross support. This should be printed out of white filament which should result in a translucent shade that works well. The present version of ringshade.stl file should be inverted as the “top” of the shade should be printed on the base of the print area.



Distance Sensor Holder

The distances sensor allows the robot to use standard HC-SR04 ultrasonic distance sensors. It also has fittings for three light sensors which can be pressed in from the front and then their wires can be accessed from the rear.

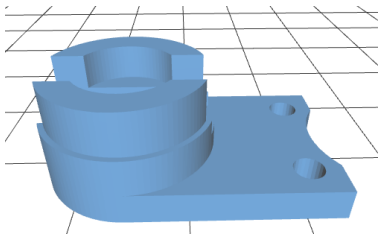
Distancesensorholder.stl: this can hold a distance sensor and three light sensors. It is fitted on the base of the robot.



Rear Pixel

You can add a rear-pixel to your robot if you like. This can be useful when using cameras to track the robot as it allows you to determine robot orientation. You use the same shade as for the single pixel.

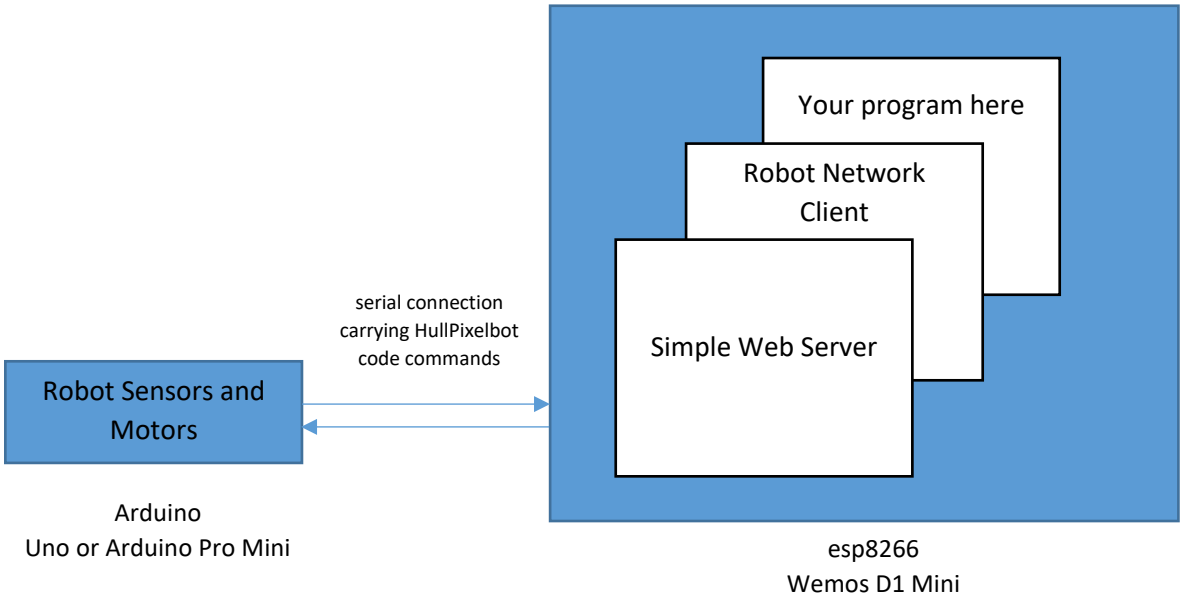
shadebase.stl: is bolted to the rear of the robot and holds a single neopixel



You can find the repository for this code here:

`https://github.com/HullPixelbot/Hardware`

Robot Software Architecture



The HullPixelbot platform uses two microcontrollers to control the robot and allow the robot to be controlled via a network connection.

An Arduino is used to control the robot sensors and motors. An esp8266 is used to provide network connectivity. The microcontrollers communicate via a serial connection which carries commands from the network connection which are to be performed by the robot.

Robot Sensors and Motors

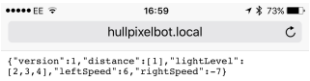
This software runs inside the Arduino which is directly connected to the motors and sensors on the robot. It contains an interpreter for the HullPixelbot Code language which can be used to instruct the robot. HullPixelbot Code statements can be executed immediately or stored inside the Arduino to provide autonomous robot behaviours. It can be built for a variety of Arduino based devices. You can find the repository for this code here:

`https://github.com/HullPixelbot/RobotSensorsAndMotors`

You can find full documentation for the HullPixelbot Code language here:

`https://github.com/HullPixelbot/HullPixelbotCode`

Simple Web Server



This software runs inside the esp8266 and implements a web server that can be used to send commands to the robot. It provides mDNS to enable discovery on a local area network. This means that Apple devices and those using Bonjour network discovery can directly access the robot.

This software requires configuration: you will have to set the name of your WiFi network and the password for that network in the program code. You can also optionally set a different name for the robot so that you can have multiple robot hosts on the same local network.

The server allows you to use query strings to send commands to the robot:

```
http://hullpixelbot.local?HullPixelbot=MF100
```

This would send the command “MF100” to the robot controller, causing the robot to move forward 100 steps. The server also returns a response in the form of a json encoded set of reading values. At present these are not mapped onto the robot sensors, but the code is included to illustrate how data values could be passed back to a caller.

A more complex web server, providing a RESTful interface to the robot commands, would be a lovely thing to have. Perhaps you might like to use this platform as the basis of such a development.

You can find the repository containing this code here:

```
https://github.com/HullPixelbot/SimpleWebServer
```

Robot Network Client

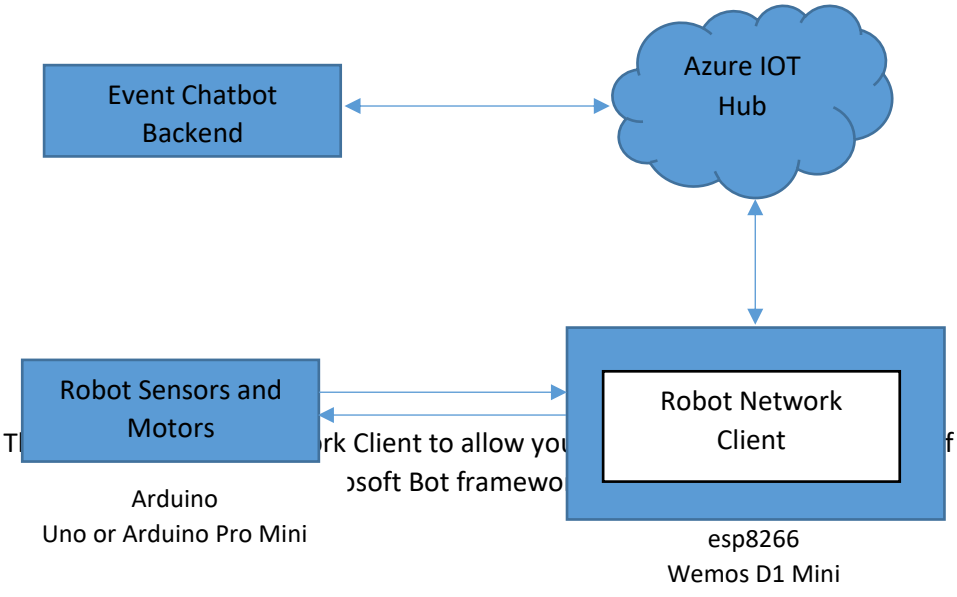
This software runs inside the esp8266 to provide an MQTT client for an Azure IOT Hub. It enables the robot to be controlled by applications running in the cloud. MQTT messages are decoded by the software and delivered to the motor controller via the serial connection. Messages received from the robot are sent into the Azure IOT Hub as MQTT messages.

Note that to use this you will require an Azure account and have set up an Azure IOT Hub. Full instructions will be posted later.

You can find the repository containing this code here:

```
https://github.com/HullPixelbot/RobotNetworkClient
```

Azure Event Manager



Th rk Client to allow you f users controlling robots via a osoft Bot framewo

Note that to use this software you will require an Azure account and have set up an Azure IOT Hub. You will also have to deploy the event manager as an Azure application. Full instructions will be posted later.

You can find the repository containing this code here:

```
https://github.com/HullPixelbot/Event-Chatbot-Backend
```

HullPixelbot Code Editor

This can be used to edit HullPixelbot programs and deploy them directly into the “Robots and Motors” application running as a motor controller. A later version will provide for the deployment of applications via MQTT and Azure IOT hub.

You can find the repository containing this code here:

```
https://github.com/HullPixelbot/HullPixelbotCodeEditor
```