The Hull Pixelbot and HullOS

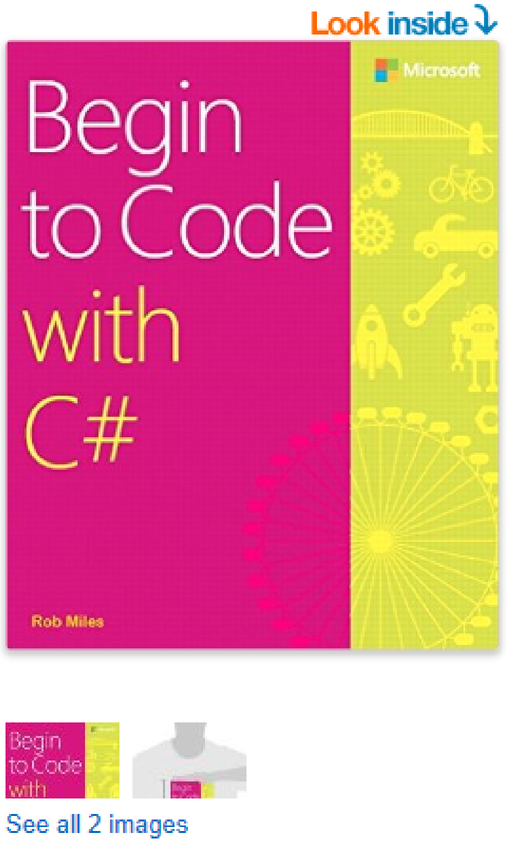
Rob Miles

www.robmiles.com

About Rob:

- Taught Computer Science at Hull University for many years
    - In charge of twisting minds and crushing dreams
- A Microsoft MVP
- Blogs at: www.robmiles.com
- Tweets at: @robmiles
- Writes books…..

Begin to Code with C#



Begin to Code with C# Paperback – 9 Sep 2016

by Rob Miles (Author)

★★★★★ 1 customer review

See all formats and editions

Kindle Edition	Paperback
£15.86	£16.69
Read with Our Free App	8 Used from £10.77
	39 New from £11.05

Want it delivered by tomorrow, 23 Nov.? Order within 2 hrs 30 mins and choose **One-Day Delivery** at checkout. Details

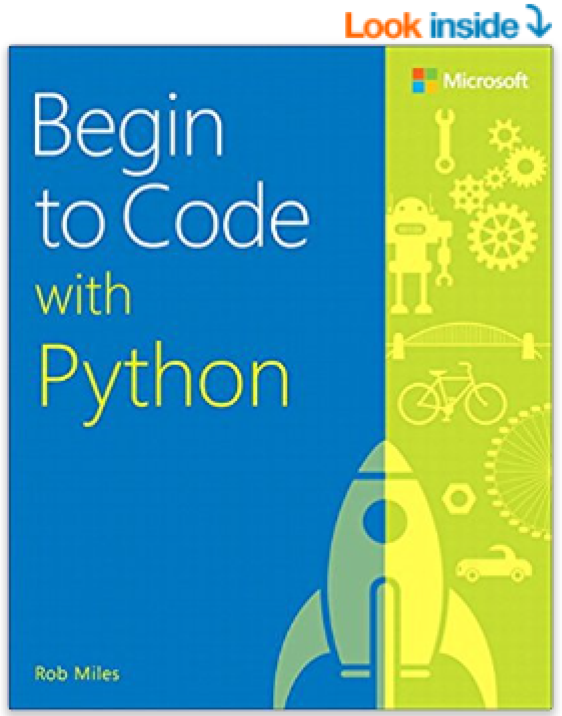
Note: This item is eligible for **click and collect.** Details

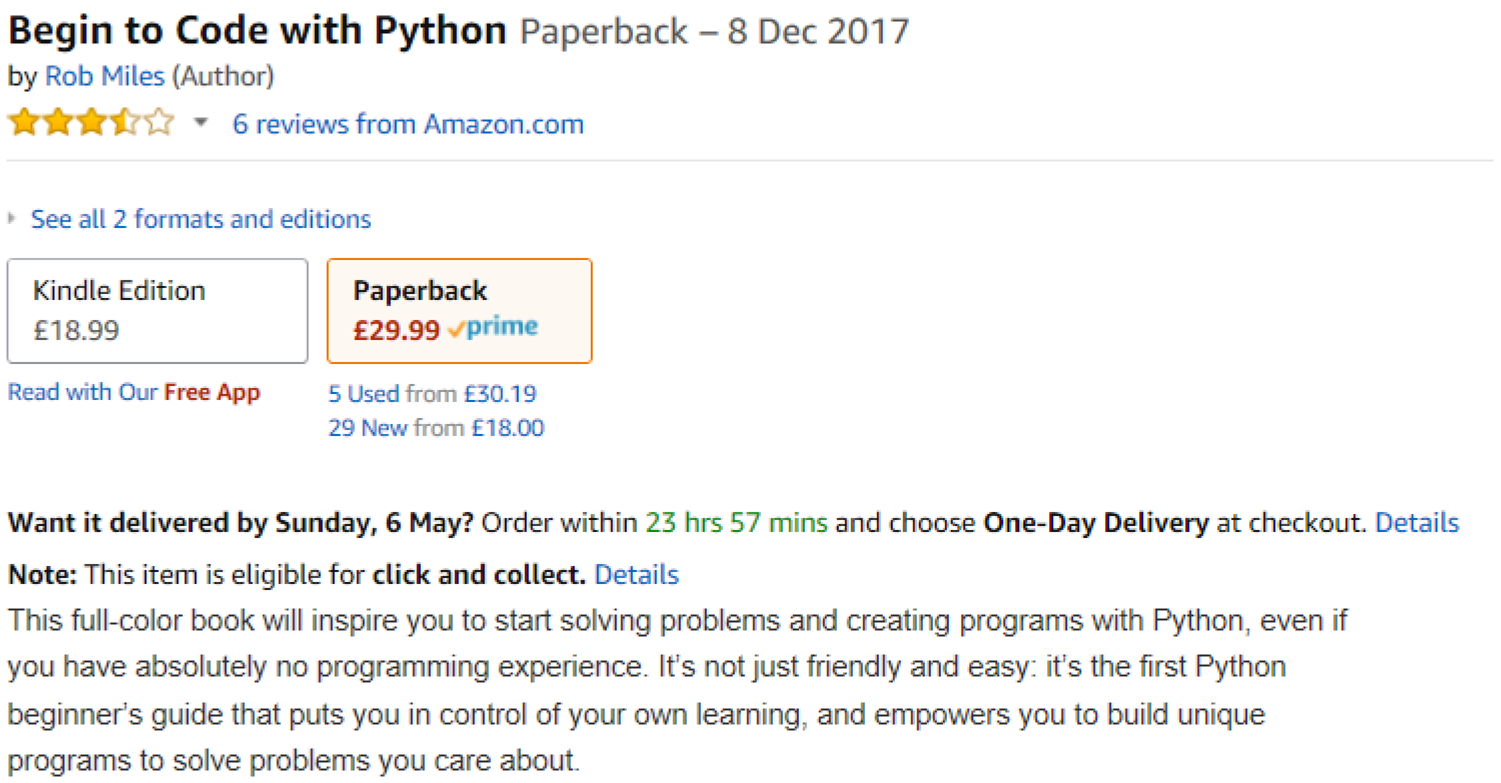
Become a C# programmer—and have fun doing it!

Start writing software that solves real problems, even if you have absolutely no programming experience! This friendly, easy, full-color book puts you in total control of your own learning, empowering you to build unique and useful programs. Microsoft has completely reinvented the beginning programmer's tutorial, reflecting deep research into how today's beginners learn, and why other books fall short. *Begin to Code with C#* is packed with innovations, from its "Snaps" prebuilt operations to its "Make Something

Read more

Begin to Code with Python





Begin to Code with Python Paperback – 8 Dec 2017
by Rob Miles (Author)
★★★½☆ 6 reviews from Amazon.com

See all 2 formats and editions

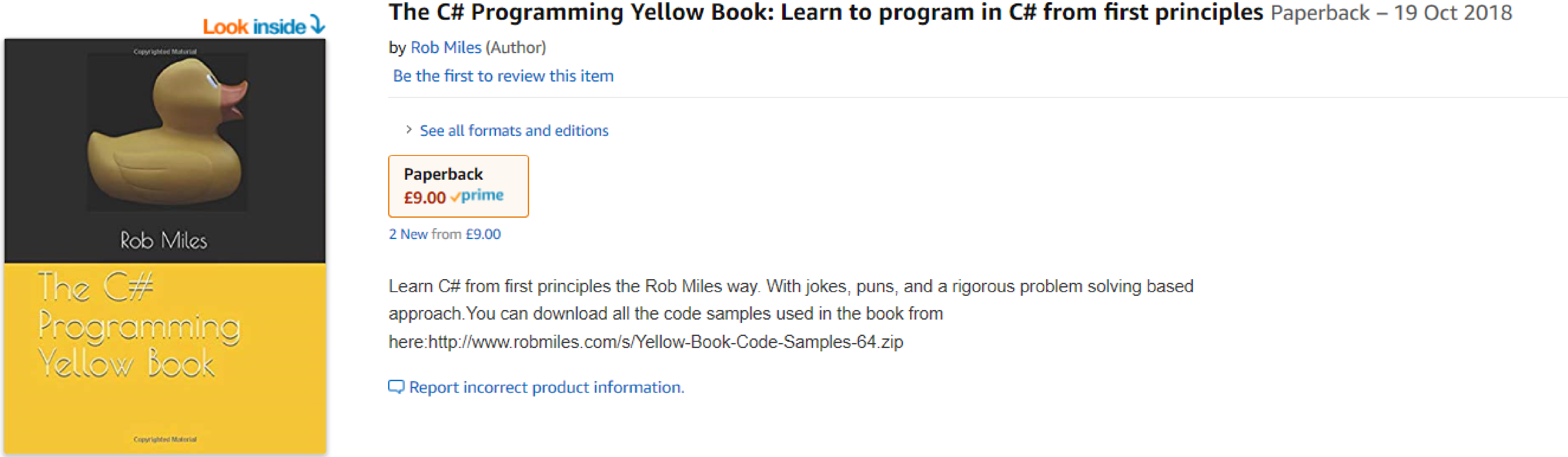
Kindle Edition	Paperback
£18.99	**£29.99** prime
Read with Our **Free App**	5 Used from £30.19 29 New from £18.00

Want it delivered by Sunday, 6 May? Order within 23 hrs 57 mins and choose **One-Day Delivery** at checkout. Details

Note: This item is eligible for **click and collect.** Details

This full-color book will inspire you to start solving problems and creating programs with Python, even if you have absolutely no programming experience. It's not just friendly and easy: it's the first Python beginner's guide that puts you in control of your own learning, and empowers you to build unique programs to solve problems you care about.

C# Yellow Book



The C# Programming Yellow Book: Learn to program in C# from first principles Paperback – 19 Oct 2018

by Rob Miles (Author)

Be the first to review this item

› See all formats and editions

Paperback
£9.00 prime

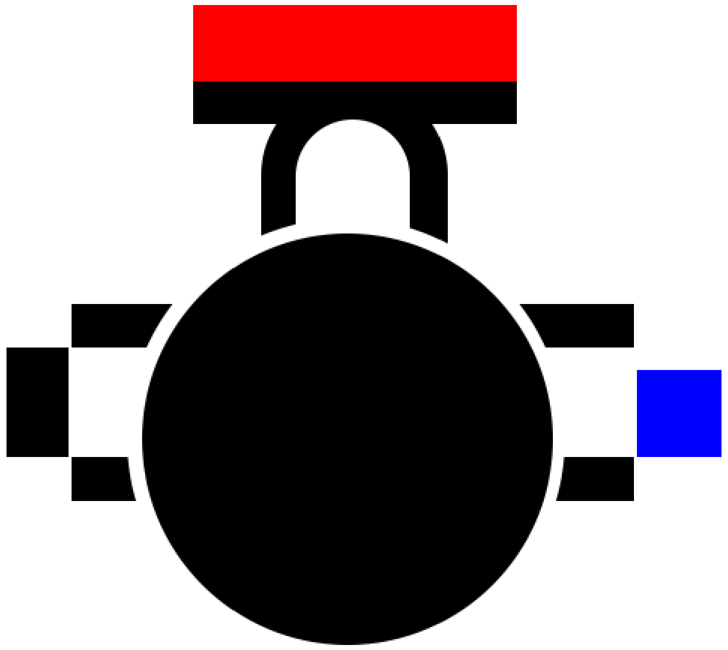
2 New from £9.00

Learn C# from first principles the Rob Miles way. With jokes, puns, and a rigorous problem solving based approach.You can download all the code samples used in the book from here:http://www.robmiles.com/s/Yellow-Book-Code-Samples-64.zip

Report incorrect product information.

Overview

- What is the Hull Pixelbot?
- HullOS – an embedded robot operating system
- Making a robot with two brains – adding a network co-processor
- Azure IoT Hub and MQTT
- Creating a web based HullOS code editor



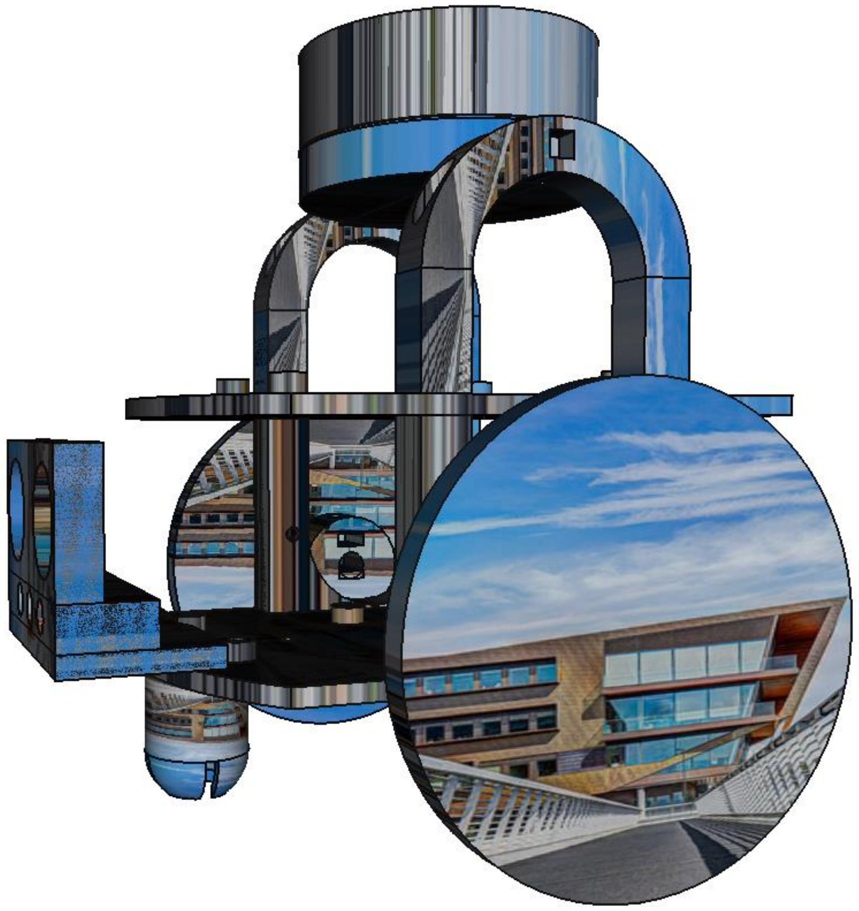
What is the Hull Pixelbot?

Hull Pixelbot project

- Flexible
- Cheap to make
- Open source
- Fun to build
- Arduino based
- Easy to program
- Extensible
- Connectable

A flexible robot

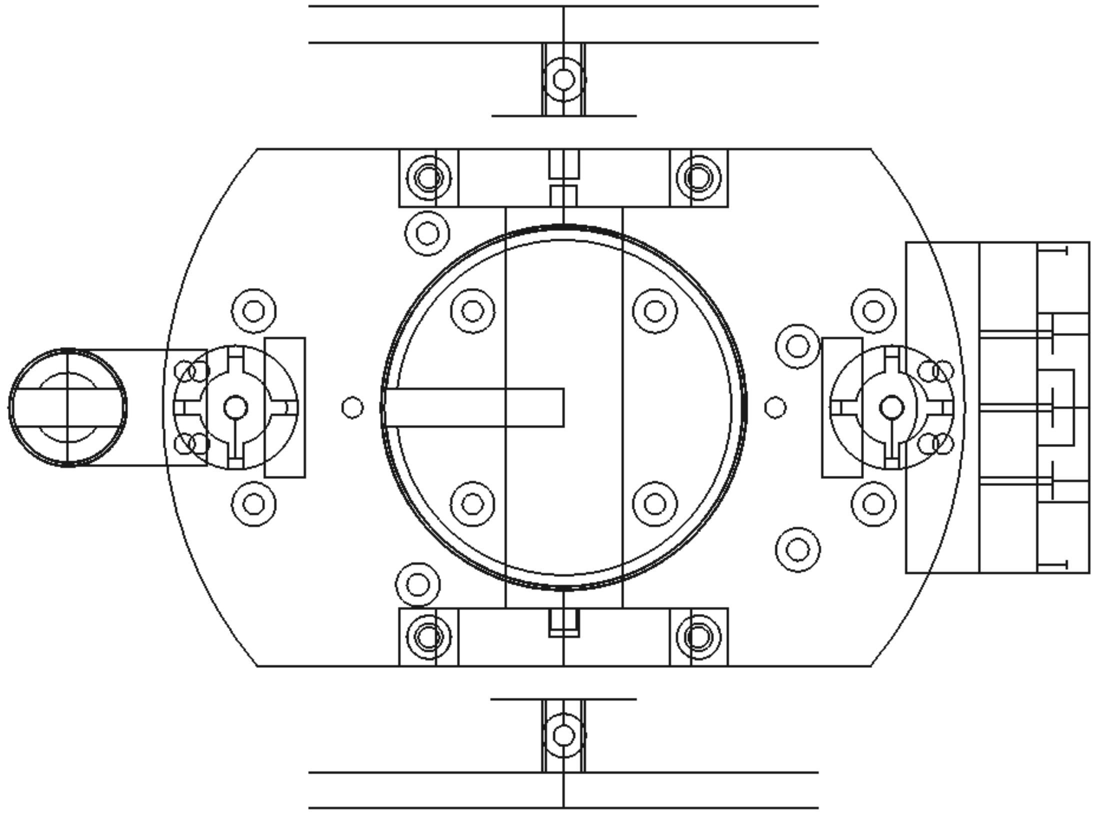
- Can be controlled by an Arduino device
  - I use the Arduino Uno or Arduino Pro-Mini
- Uses stepper motors for movement
  - Slow but very precise
- Has a coloured pixel
  - Allows you to give your robot a personality
- Has a distance sensor
  - Allows the robot to react to its environment
- Other sensors can be added as you fancy



Cheap to make

- Arduino processor
  - Less than five pounds
- Cheap stepper motor
  - Around one-fifty each
- Cheap pixel ring
  - Around one-fifty
- Distance sensor, battery holder, cables and nuts and bolts add around four pounds
- You can get the electronics for around ten pounds our so: www.aliexpress.com

Open Source

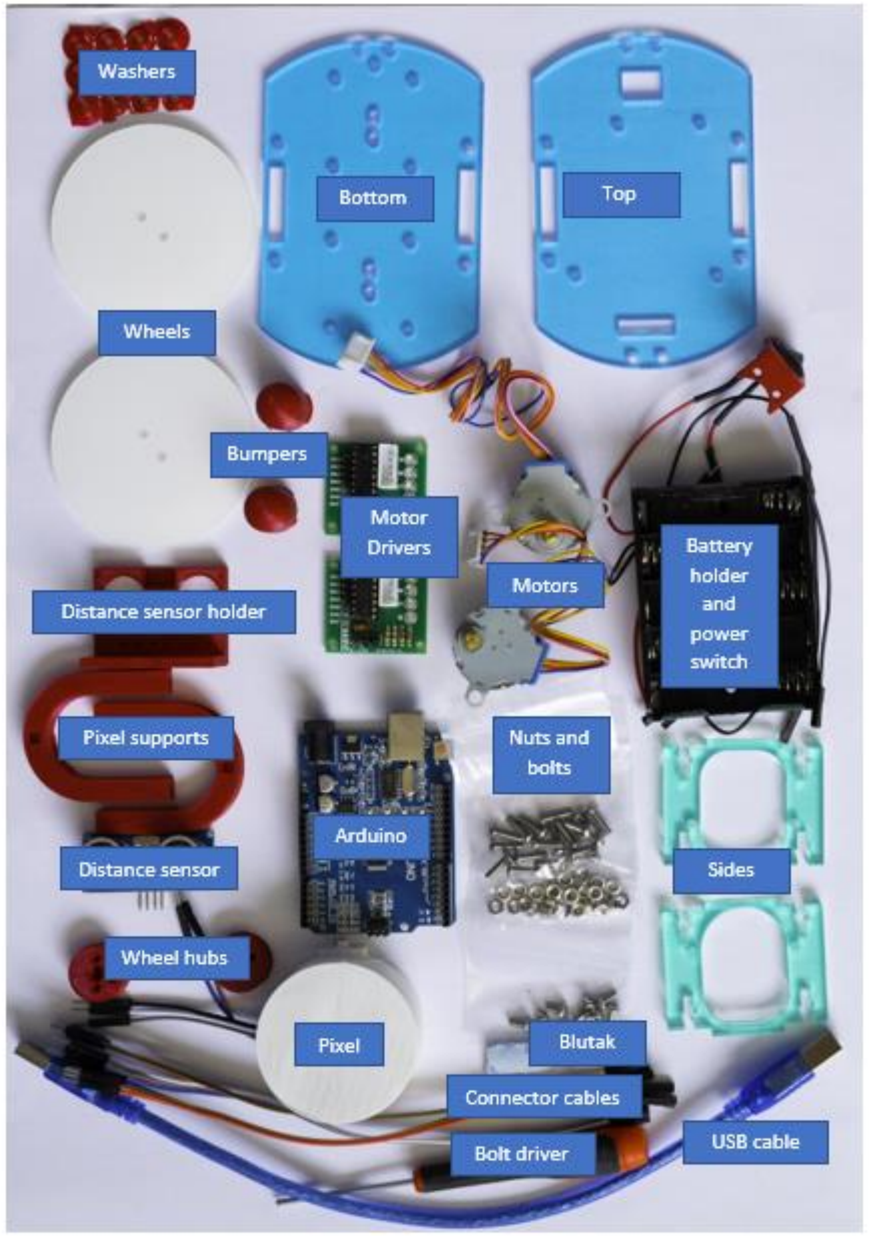


- The designs for the robot and controlling software are all open source
- Everything is published on GitHub

https://github.com/HullPixelbot

Fun to build

- Full construction notes are on GitHub
- I can supply laser cut and 3D printed elements
    - Or you can make them yourself
- You can even design your own robot chassis and just use the software if you wish
- I also do “Build a Robot in a Day” events if you fancy signing up for one



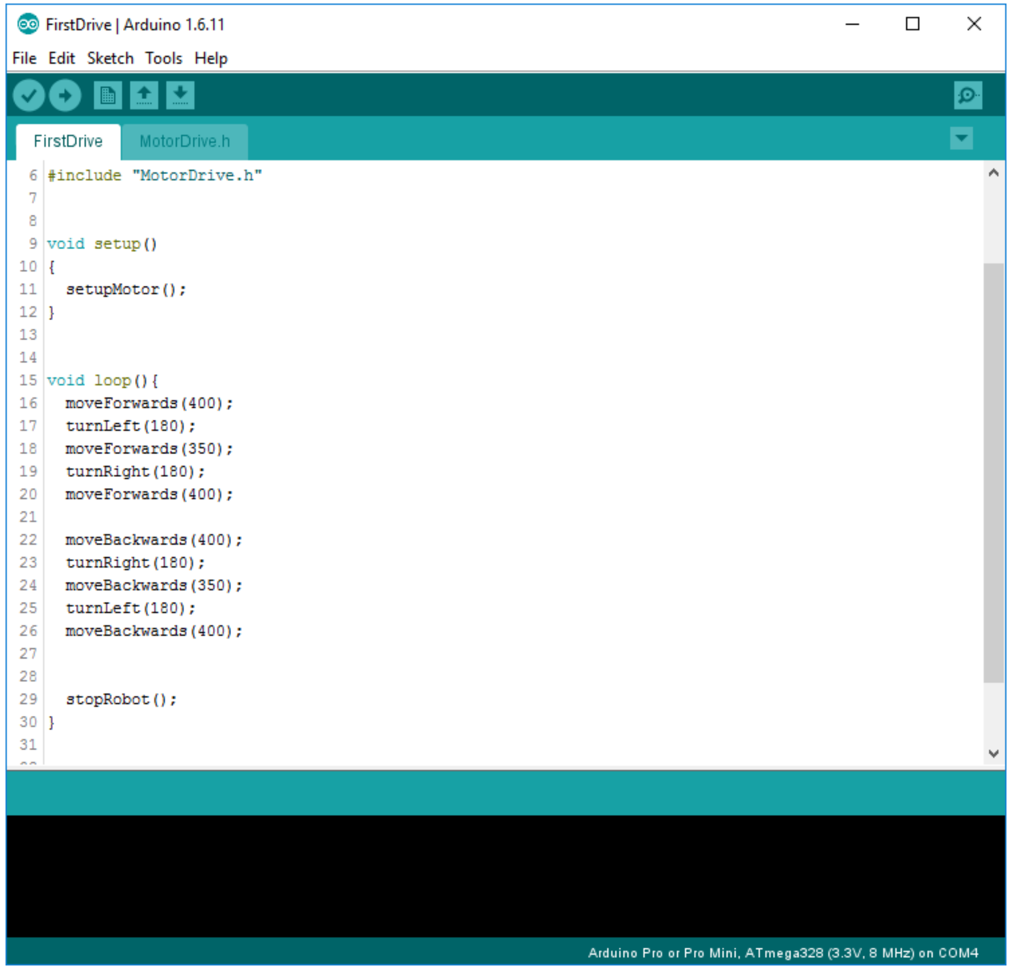
The Arduino Uno

- Arduino Uno
  - Advantages:
    - Really cheap
    - Easy to write code
    - Plenty of i/o – both analogue and digital
    - A proper embedded device
      - code runs directly on the hardware
  - Disadvantages
    - Not that powerful
    - Very limited program and data space (32K and 2K)
    - No networking ability built in
      - you have to use serial connections to transfer information

  - The Uno is fantastic for simple, disconnected devices but is no good for anything that you'd like to connect to the outside world.

Easy to Program?

- You can write programs in C++ using the Arduino environment
  - Programs are downloaded into the Arduino via the serial port and persisted in EEPROM
- It is very easy to get simple things to happen, for example lights and movement, but more tricky when you want to do several things at once

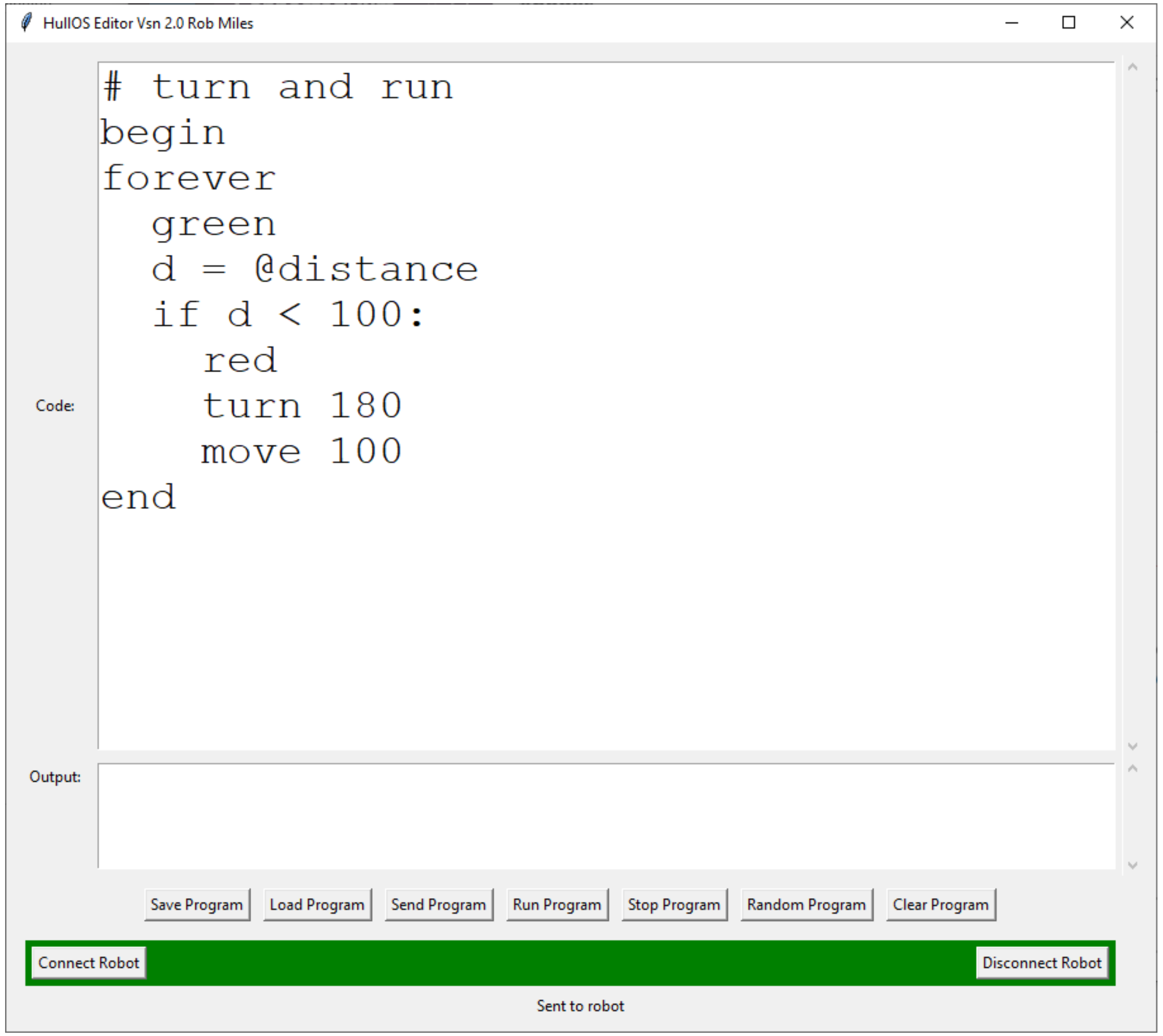


```
#include "MotorDrive.h"


void setup()
{
  setupMotor();
}


void loop(){
  moveForwards(400);
  turnLeft(180);
  moveForwards(350);
  turnRight(180);
  moveForwards(400);

  moveBackwards(400);
  turnRight(180);
  moveBackwards(350);
  turnLeft(180);
  moveBackwards(400);


  stopRobot();
}
```

Arduino Pro or Pro Mini, ATmega328 (3.3V, 8 MHz) on COM4

Easy to program

- HullOS provides a simple environment that can be used to create programs for the robot
- The program code is interpreted and executed on the robot itself
- It runs on a multi-tasking platform



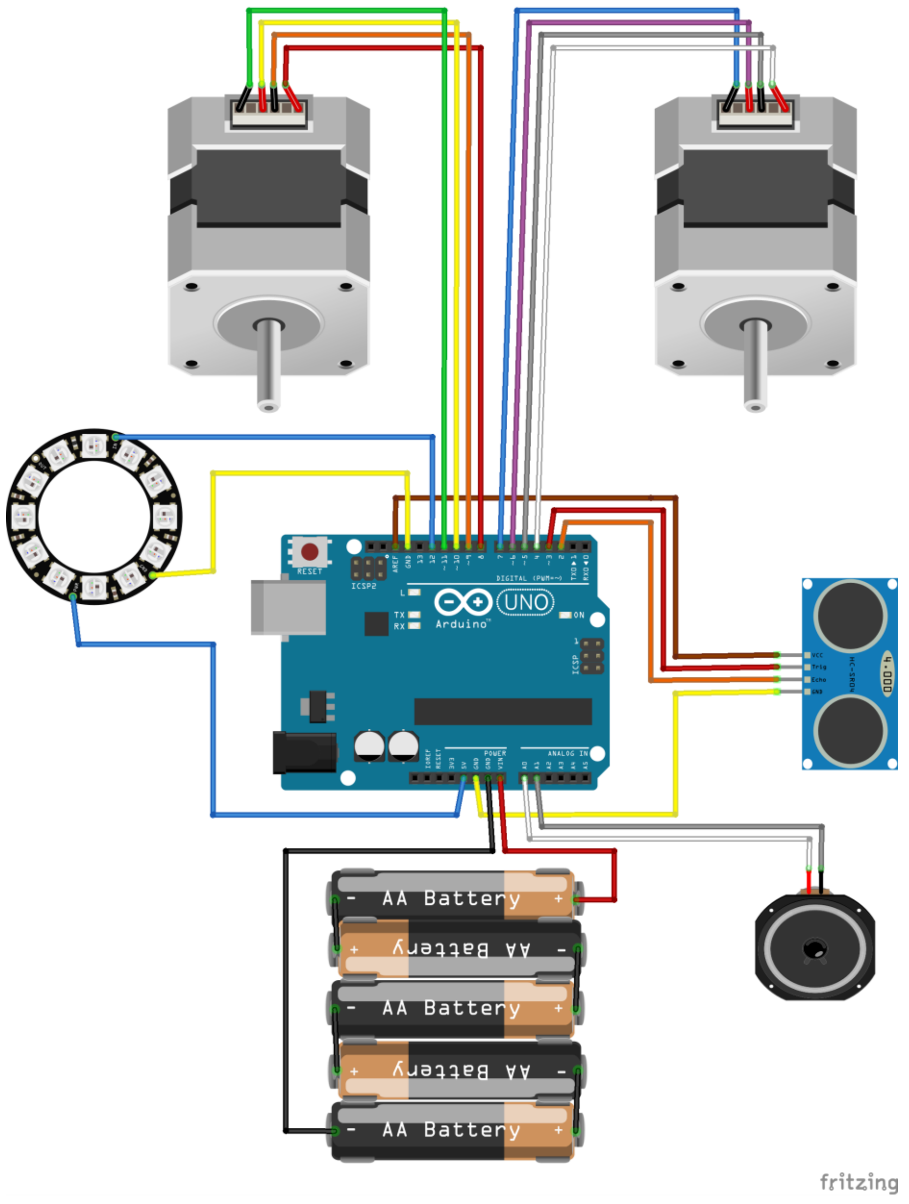
```
# turn and run
begin
forever
  green
  d = @distance
  if d < 100:
    red
    turn 180
    move 100
end
```

Sent to robot

Demo 1: Talking to a robot

Robot extensibility

- The Arduino controls the motors, pixel and speaker
- It also receives data from the distance sensor and the serial port
  - There are six analogue/digital connections remaining available
- You can use this platform as a robot controller that underpins a more powerful device



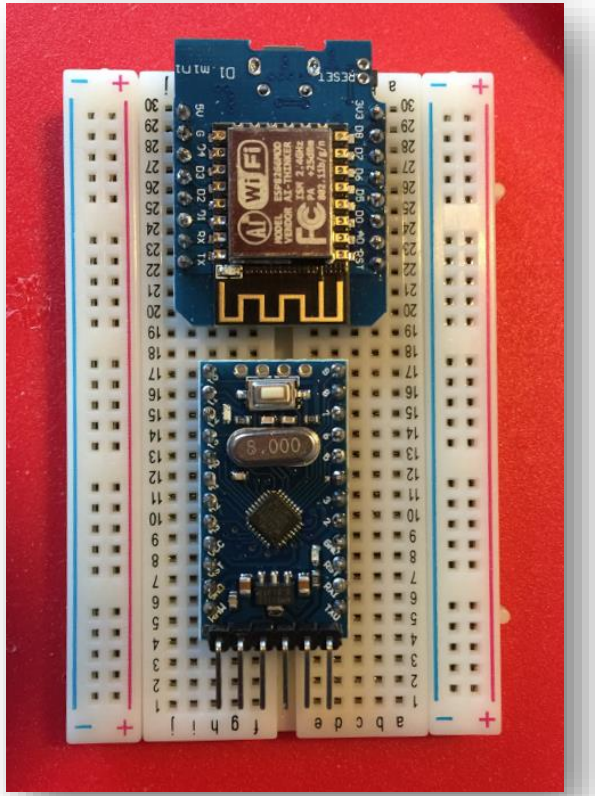
HullOS extensibility

```
int readRandom()
{
    return random(1, 13);
}

struct reading randomReading = { "random", readRandom };

#define NO_OF_HARDWARE_READERS 4

struct reading * readers[NO_OF_HARDWARE_READERS] = { &distance, &light, &moving, &randomReading };
```

- HullOS has been designed to be easy to extend
- We can add new sensors and outputs very easily

Connectable

- The Arduino itself is not a very connectable device
    - It only has a serial port for external connectivity
- But it is easy to connect the Arduino robot controller to another device that provides a connection
- That device can then offload all the robot control to the Arduino



HullOS – an embedded robot operating system

HullOS

- I wanted to use a Hull Pixelbot to teach programming
  - It turns out that writing C++ is not a great starting point
- So, I’ve invented my own embedded Operating System
  - It is called HullOS
  - It implements an embedded scripting language which is quite fun

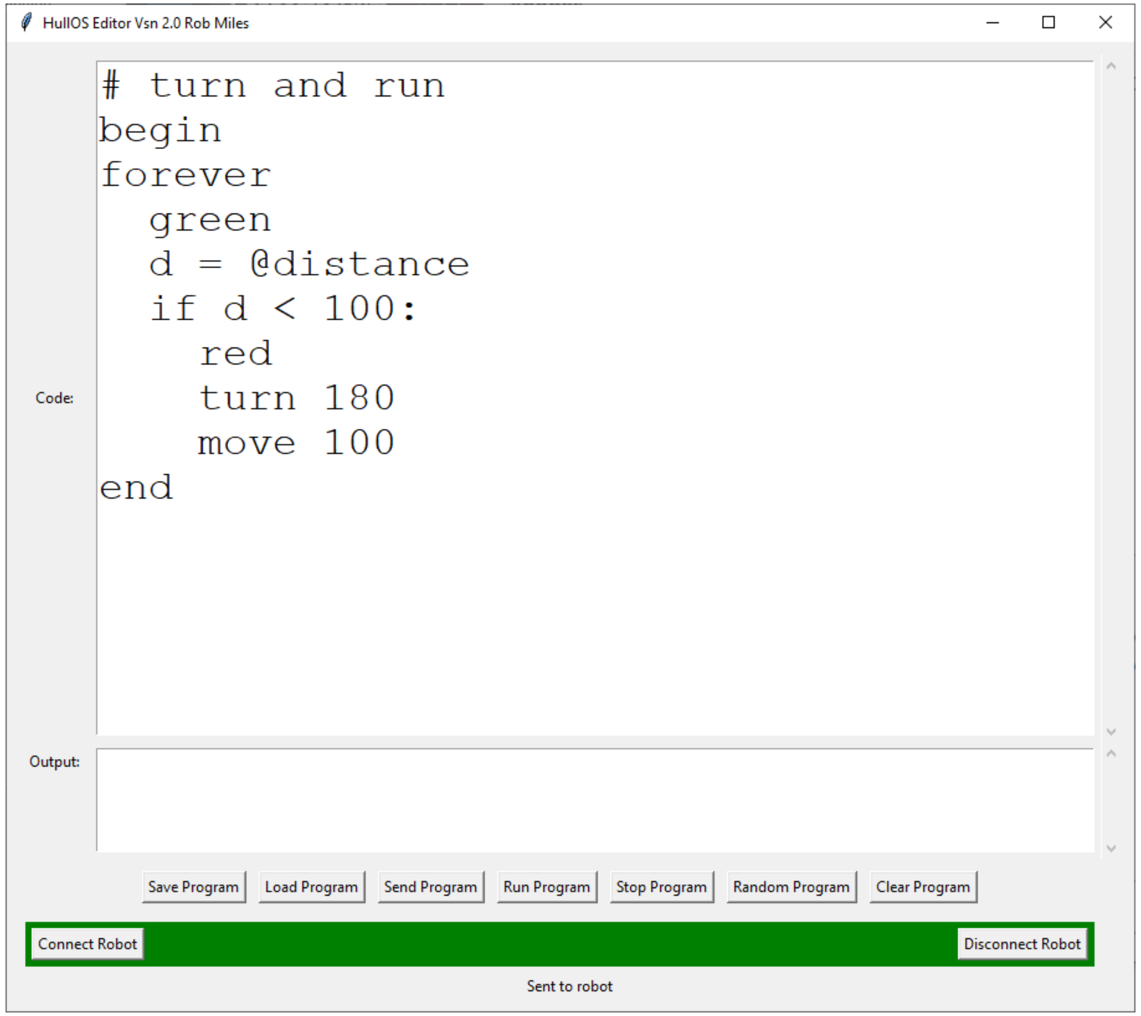
HullOS scripting

- The HullOS scripting language is a bit like Python
- It is compiled and interpreted on the robot platform
  - We send the program into the robot in pure text via the serial port
- The robot retains the compiled program in EEPROM and runs it when it is powered on

```
# avoid obstacles
begin
forever
  green
  move
  d = @distance
  if d < 100:
    red
    turn 90
end
```

Sending programs to the robot

- I've created a terminal program in Python that does the downloading of the programs and runs them
- There's also a C# version
- You can type in the programs using any terminal program

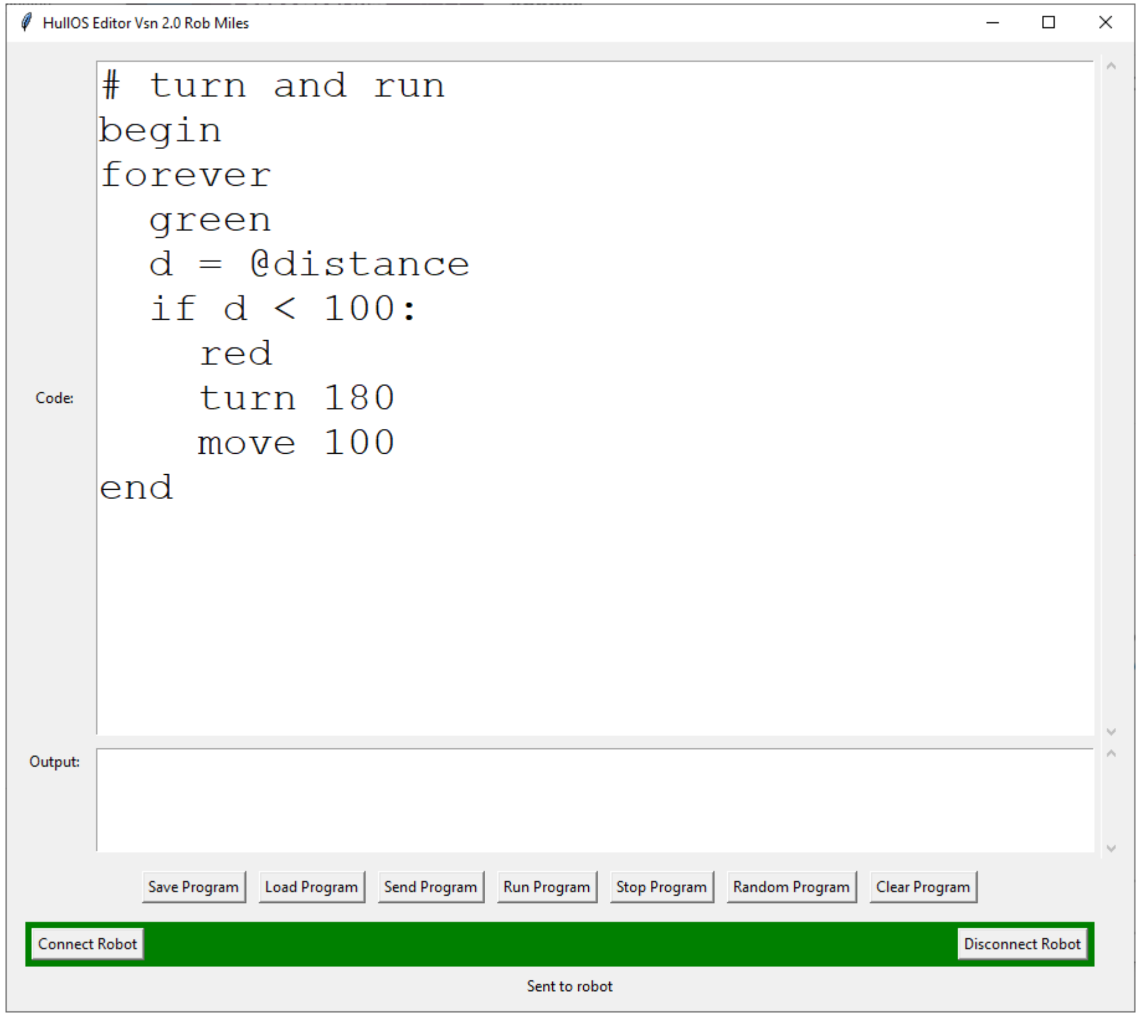


```
# turn and run
begin
forever
  green
  d = @distance
  if d < 100:
    red
    turn 180
    move 100
end
```

Sent to robot

Sending programs to the robot

- We can send programs to the robot down the serial port
- HullOS converts the programs into an intermediate code that is interpreted
- The intermediate code is stored in EEPROM in the Arduino



```
# turn and run
begin
forever
  green
  d = @distance
  if d < 100:
    red
    turn 180
    move 100
end
```

Sent to robot

HullOS Intermediate Language

- The Intermediate Language is stored inside the robot
- Each command is a two letter code
  - Command family
  - Command option
- This is interpreted when the program runs
- It is all powered by switch statements…

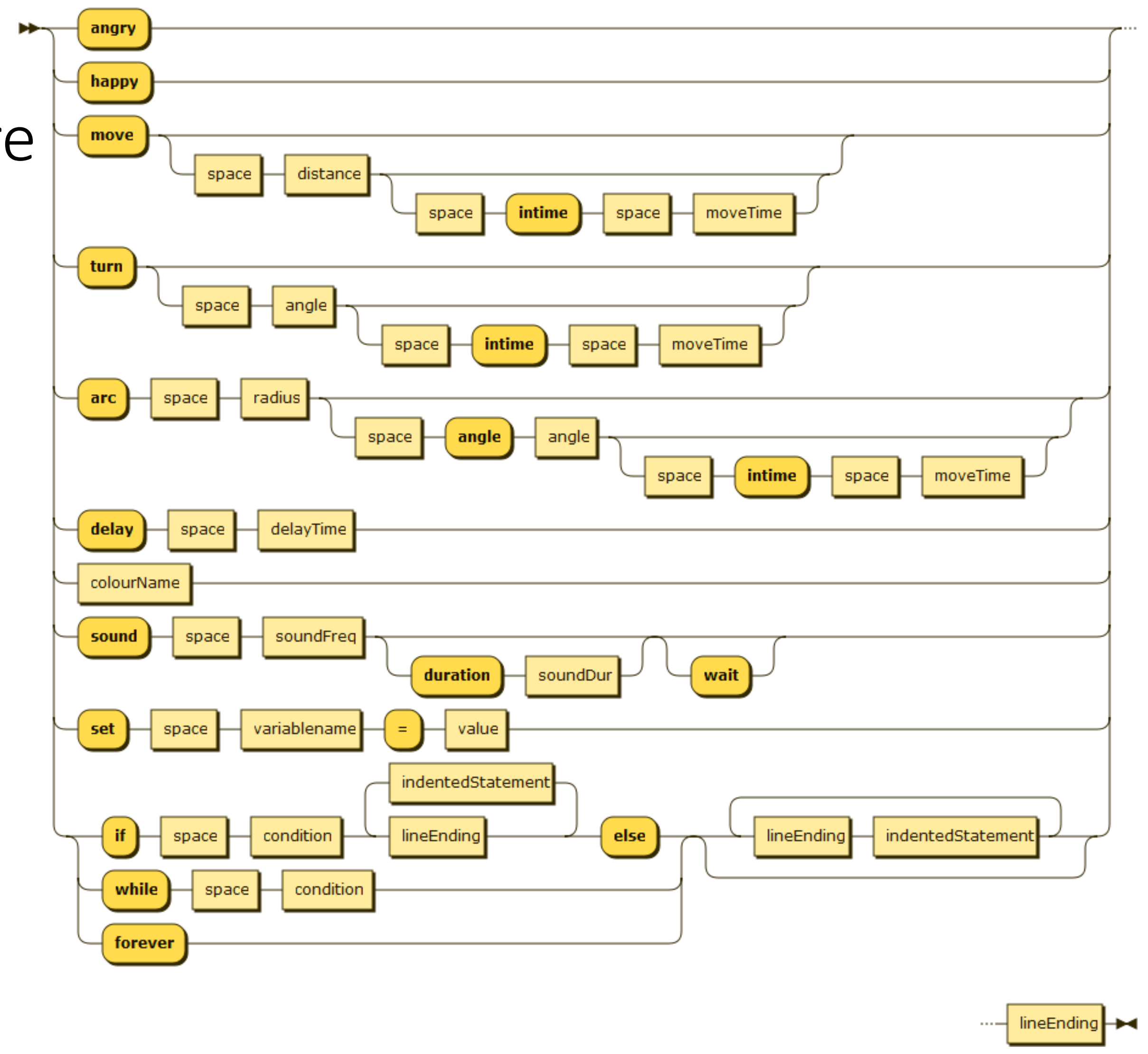
```
CLl1
PNg
MF20000
VSd=@distance
CFd<100,l3
PNr
MR90
CA
CLl3
CJl1
CLl2
```

HullOS Intermediate Language

```
CLl1            << Loop Label
PNg             << Go green
MF20000         << Move a long way
VSd=@distance   << Load variable d with distance value
CFd<100,l3      << If d is less than 100 jump to l3
PNr             << Go red
MR90            << Rotate 90 degrees
CA              << Wait for the rotate to complete
CLl3            << Jump destination if condition fails
CJl1            << Jump the top of the loop
CLl2            << Exit label for loop used by break
```

High Level Languages and Magic

- I want to make it clear that there is nothing "magical" about how programming languages work
- HullOS script uses the same fundamental principles as all languages
- It is very easy to add new high level and low level language features

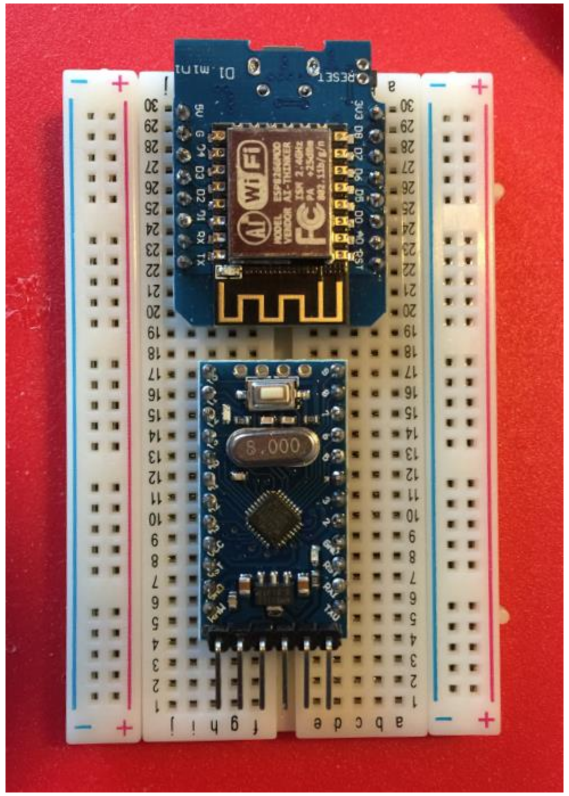


Demo 2: Inside HullOS

Making a robot with two brains – adding a network co-processor

The Robot with two brains….

- Arduino Pro and ESP8266
  - The HullPixelbot can use two processors
    - Arduino Pro mini for the input/output and motor control
    - Wemos D1 mini for the connectivity
  - This is a great way to create i/o heavy devices
    - Use a serial connection to pass commands between the two
  - For most simple systems you only really need a single device
    - But the Arduino Pro mini only costs around a pound….



The esp8266 is an awesome chip….

- Lots of WiFi options
  - WiFi client over a serial port
  - Fully programmable in C++ just like the Arduino
  - WiFi access point and web server
  - Support for UDP, TCP, secure sockets and mDNS
  - Very easy to use with many examples
- Making a connected client device
  - Lots of ways to do this
  - We're going to use MQTT but you can use it as a web server, or even a WiFi access point (or both)
- I use the Wemos platform – around two pounds fifty a pop…

WiFi and the esp8266

- Lots of WiFi options
  - WiFi client over a serial port
  - Fully programmable in C++
  - WiFi access point and web server
  - Support for UDP, TCP, secure sockets and mDNS
  - Very easy to use with many examples
- Making a connected client device
  - Lots of ways to do this
  - Web server, web sockets, MQTT, LAN, Access Point

An esp8266 as a web server

- Create an embedded web server on a network:
  1. Select the Wemos D1 R2 and D2 mini platform
  2. Select the ESP8266WebServer>HelloServer example
  3. Set the SSID and the password in the code
  4. Deploy the program
  5. Connect via a terminal to view server output
  6. Connect via device on same subnet: esp8266.local using MDNS
- C++ methods are fired to deal with web requests
- This makes it possible to use web protocols to do just about anything with the device

An esp8266 as an access point

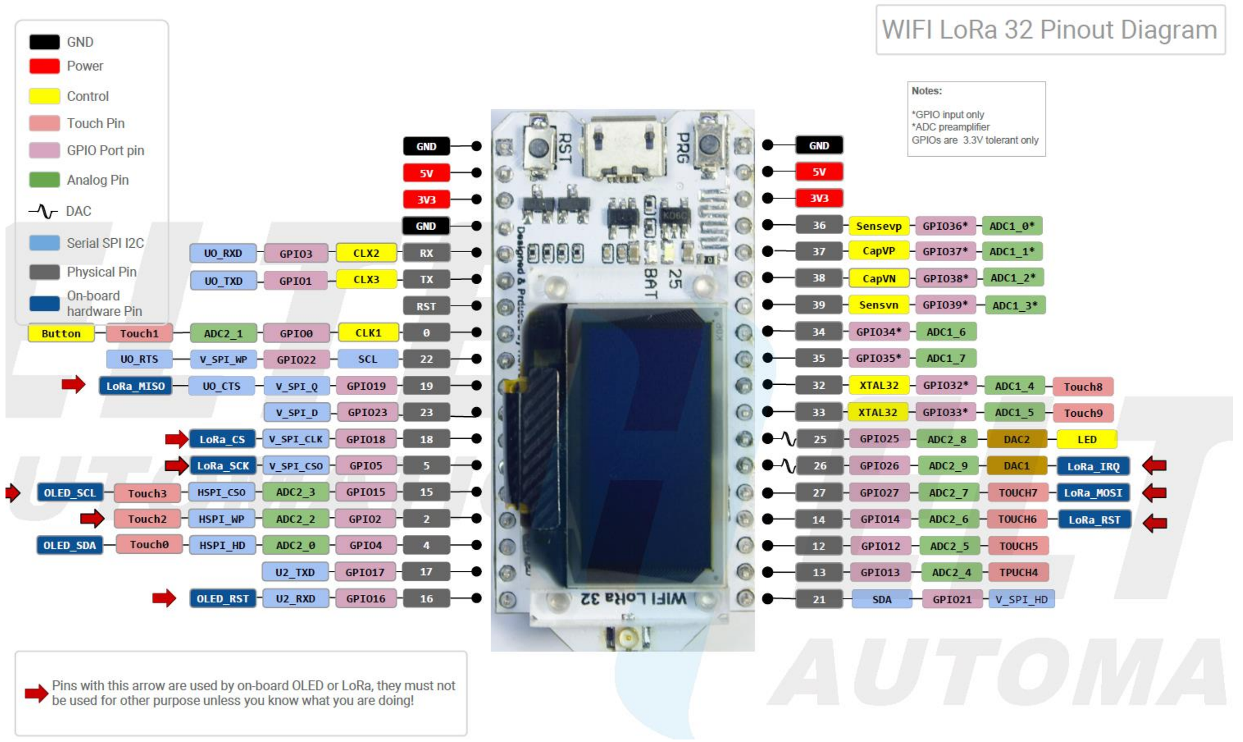
- Create an embedded web server on its own access point:
  1. Select the Wemos D1 R2 and D2 mini platform
  2. Select the ESP8266WiFi>WiFiAccessPoint example
  3. Set the SSID and the password in the code
  4. Deploy the program
  5. Connect via a terminal to view server output
  6. Connect a device to the access point. Browse: http://192.168.4.1
- This is a popular way to perform initial device configuration
  - Device hosts a web site that allows the entry of the WiFi parameters which are then stored in EEPROM for future use

Esp8266 as an access point

- Only really supports one client at a time
- Can use websockets to communicate with the device
- The esp8266 also contains an internal file store that you can use to hold html files and other resources
- There is support for over the air (OTA) updates via WiFi
- Will support HTTPS connections too

Enter the ESP32

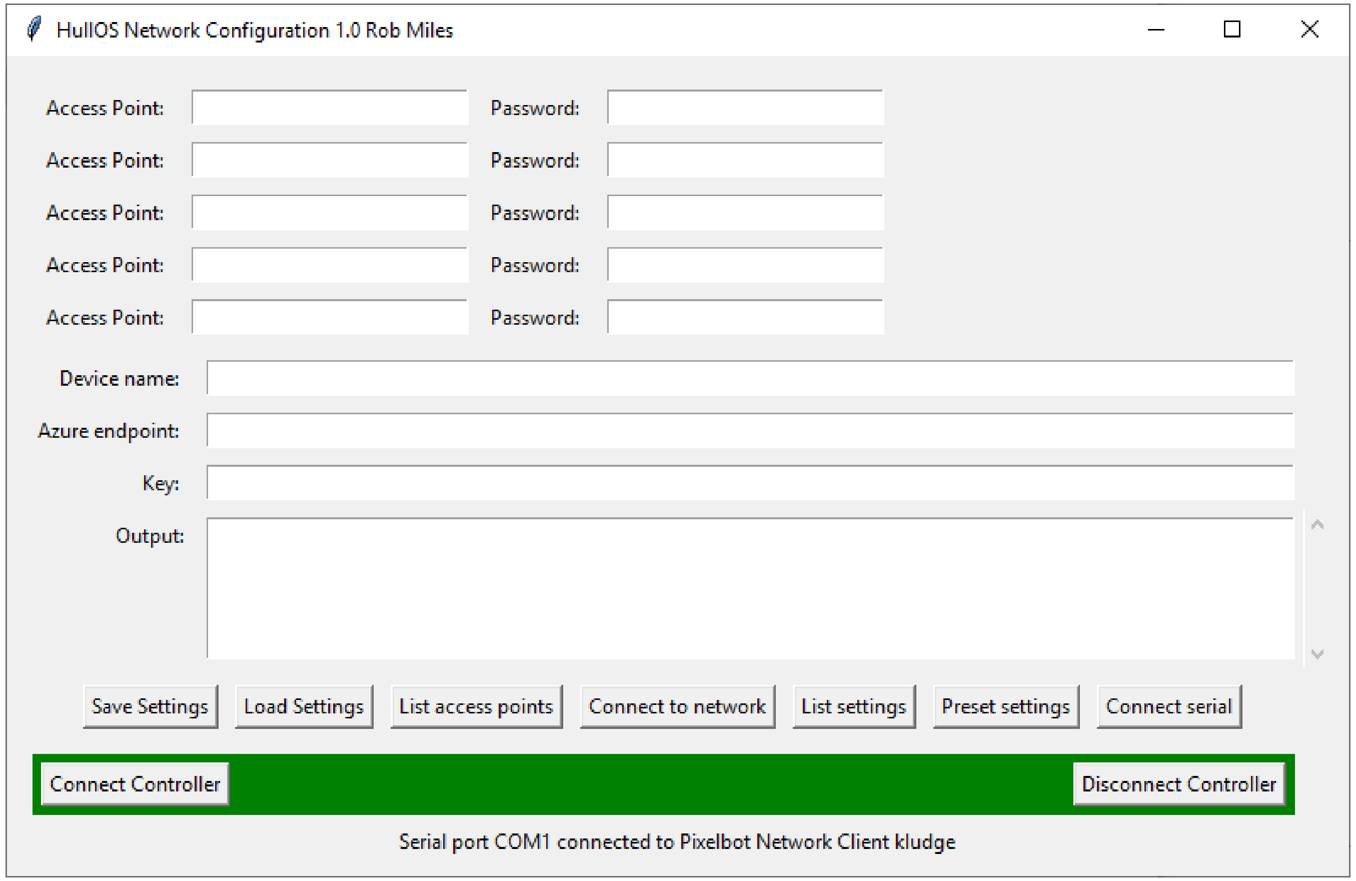
- The company that made the ESP8266 has now made its successor - the ESP32
- This is a dual core device with 16M of RAM clocked at 240MHz
- It costs around a fiver
- You can program it with the Arduino IDE or Python
- The Heltec version costs a bit more (12 pounds) but includes an OLED screen and a LoRa (Low powered Radio) device



WIFI LoRa 32 Pinout Diagram
GND
Power
Control
Touch Pin
GPIO Port pin
Analog Pin
DAC
Serial SPI I2C
Physical Pin
On-board hardware Pin
Notes:
*GPIO input only
*ADC preamplifier
GPIOs are 3.3V tolerant only
GND 5V 3V3 GND
U0_RXD GPIO3 CLX2 RX
U0_TXD GPIO1 CLX3 TX
RST
Button Touch1 ADC2_1 GPIO0 CLK1 0
U0_RTS V_SPI_WP GPIO22 SCL 22
LoRa_MISO U0_CTS V_SPI_Q GPIO19 19
V_SPI_D GPIO23 23
LoRa_CS V_SPI_CLK GPIO18 18
LoRa_SCK V_SPI_CS0 GPIO5 5
OLED_SCL Touch3 HSPI_CS0 ADC2_3 GPIO15 15
Touch2 HSPI_WP ADC2_2 GPIO2 2
OLED_SDA Touch0 HSPI_HD ADC2_0 GPIO4 4
U2_TXD GPIO17 17
OLED_RST U2_RXD GPIO16 16
GND 5V 3V3
36 Sensevp GPIO36* ADC1_0*
37 CapVP GPIO37* ADC1_1*
38 CapVN GPIO38* ADC1_2*
39 Sensvn GPIO39* ADC1_3*
34 GPIO34* ADC1_6
35 GPIO35* ADC1_7
32 XTAL32 GPIO32* ADC1_4 Touch8
33 XTAL32 GPIO33* ADC1_5 Touch9
25 GPIO25 ADC2_8 DAC2 LED
26 GPIO26 ADC2_9 DAC1 LoRa_IRQ
27 GPIO27 ADC2_7 TOUCH7 LoRa_MOSI
14 GPIO14 ADC2_6 TOUCH6 LoRa_RST
12 GPIO12 ADC2_5 TOUCH5
13 GPIO13 ADC2_4 TPUCH4
21 SDA GPIO21 V_SPI_HD
RST PRG BAT 25
WIFI LoRa 32
Pins with this arrow are used by on-board OLED or LoRa, they must not be used for other purpose unless you know what you are doing!

Hull Pixelbot Network Client

- The Hull Pixelbot Network client program runs in a connected device
- It connection to MQTT
- It provides a serial connection that is used to configure a robot
- The client passes network messages into the Arduino to control the robot


HullOS Network Configuration 1.0 Rob Miles

Access Point:		Password:	
Access Point:		Password:	
Access Point:		Password:	
Access Point:		Password:	
Access Point:		Password:	

Device name:

Azure endpoint:

Key:

Output:

Serial port COM1 connected to Pixelbot Network Client kludge

Connecting a robot with MQTT

Message Queue Telemetry Transport

- MQTT is a way to connecting sensors to endpoints
  - It has a publish/subscribe architecture
- The communication can run over serial or WiFi and is based on a simple packet structure
- People have different opinions of how good it is, but it is very popular and also supported by the Azure IOT Hub among other people..
- It also runs (surprise surprise) on the esp8266 and ESP32
- It is a great way to create cheap, connected, sensors

Azure, MQTT and the esp8266

- MQTT PubSubClient for esp8266
  - I'm using the PubSubClient for esp8266 available at https://github.com/knolleary/pubsubclient
  - It needs to be modified for Azure:
    - Azure uses secure sockets, a different port and has larger packets
    - You can find out how to set everything up here:
    - http://www.radupascal.com/2016/04/03/esp8266-arduino-iot-hub
- Azure IoT Hub
  - The Azure IoT Hub will respond to MQTT messages
  - These can be passed on to your backend Azure applications and Azure applications can target MQTT devices

Setting up MQTT in the Wemos

```
void setup() {
    Serial.begin(9600);
    robotSerial.begin(9600);
    setup_wifi();
    client.setServer(mqtt_server, 8883);
    client.setCallback(callback);
}
```

- This is the setup function
  - It runs when the device starts
  - It starts the server running :
    - mqtt_server contains the name of the server : HullPixelbot.azure-devices.net
    - 8883 is the port number being used (this is an Azure thing)
  - It also binds a method (**callback**) to incoming MQTT messages from the server

Making the connection

```
void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    if (client.connect("RedRobot","HullPixelbot.azure-devices.net/RedRobot",
                       "SharedAccessSignature sr=HullPixelbot.azure-
         devices.net%2Fdevices%2redrobot&sig=1zsdfsweraerweeY7lp4Kc1x%2B%2FhVZ7apgGWQQ%3D&se=1")){
      Serial.println("connected");
      client.publish("devices/RedRobot/messages/events/", “robot started");
      client.subscribe("devices/RedRobot/messages/devicebound/#");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}
```

Making the connection

```
void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    if (client.connect("RedRobot","HullPixelbot.azure-devices.net/RedRobot",
                       "SharedAccessSignature sr=HullPixelbot.azure-
          devices.net%2Fdevices%2redrobot&sig=1zsdfsweraerweeY7lp4Kc1x%2B%2FhVZ7apgGWQQ%3D&se=1")){
      Serial.println("connected");
      client.publish("devices/RedRobot/messages/events/", "        tarted");
      client.subscribe("devices/RedRobot/messages/devicebound     ;
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 second
      delay(5000);
    }
  }
}
```

The access key is created by the Azure IOT device manager
It can be given a lifetime after which the device can no longer connect

Making the connection

```
void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    if (client.connect("RedRobot","HullPixelbot.azure-devices.net/RedRobot",
                       "SharedAccessSignature sr=HullPixelbot.azure-
          devices.net%2Fdevices%2redrobot&sig=1zsdfsweraerweeY7lp4Kc1x%2B%2FhVZ7apgGWQQ%3D&se=1")){
      Serial.println("connected");
      client.publish("devices/RedRobot/messages/events/", “robot started");
      client.subscribe("devices/RedRobot/mess          /devicebound/#");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.sta
      Serial.println(" try ag
      delay(5000);
    }
  }
}
```

This statement publishes a message to the Azure IOT hub to tell the hub that the device is connected

Making the connection

```
void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    if (client.connect("RedRobot","HullPixelbot.azure-devices.net/RedRobot",
                       "SharedAccessSignature sr=HullPixelbot.azure-
       devices.net%2Fdevices%2redrobot&sig=1zsdfsweraerweeY7lp4Kc1x%2B%2FhVZ7apgGWQQ%3D&se=1")){
      Serial.println("connected");
      client.publish("devices/RedRobot/messages/events/", “robot started");
      client.subscribe("devices/RedRobot/messages/devicebound/#");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}
```

This statement subscribes to messages from the IOT hub
When a message arrives the callback function is called to deal with it

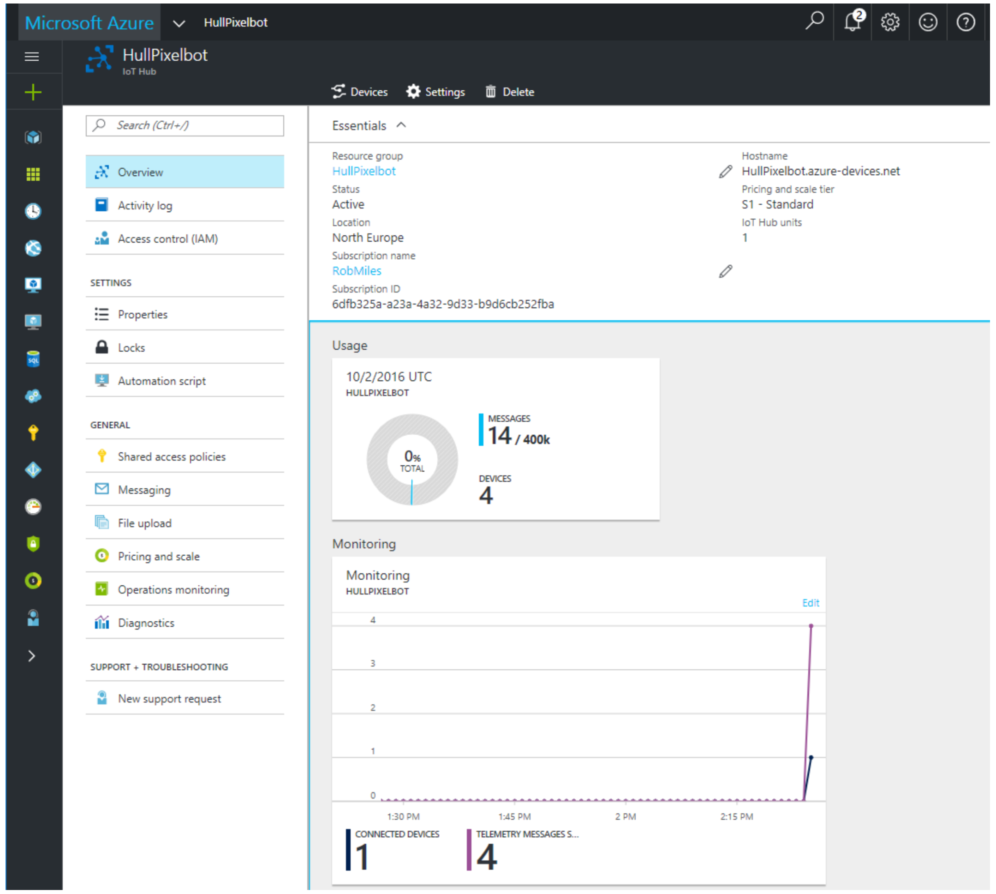
Decoding incoming messages

```
char message [200];
void callback(char* topic, byte* payload, unsigned int length) {
    int i;
    // Build a robot command
    for (i = 0; i < length; i++) {
        message[i] = (char)payload[i];
    }

    // Put the terminating character on the end of the message
    message[i] = 0;

    // Pass the command onto the motor processor
    sendRobotCommand(message);
}
```

- This method runs when the robot receives a message
- The message is passed into the Arduino that controls the robot

Azure IoT Hub

- The IoT hub will collect and manage messages from connected devices
- It exposes service points to connect other Azure applications
- You can also set up your own MQTT hub on other platforms, for example Raspberry Pi, or use one hosted by AdaFruit



Resource group
HullPixelbot
Status
Active
Location
North Europe
Subscription name
RobMiles
Subscription ID
6dfb325a-a23a-4a32-9d33-b9d6cb252fba

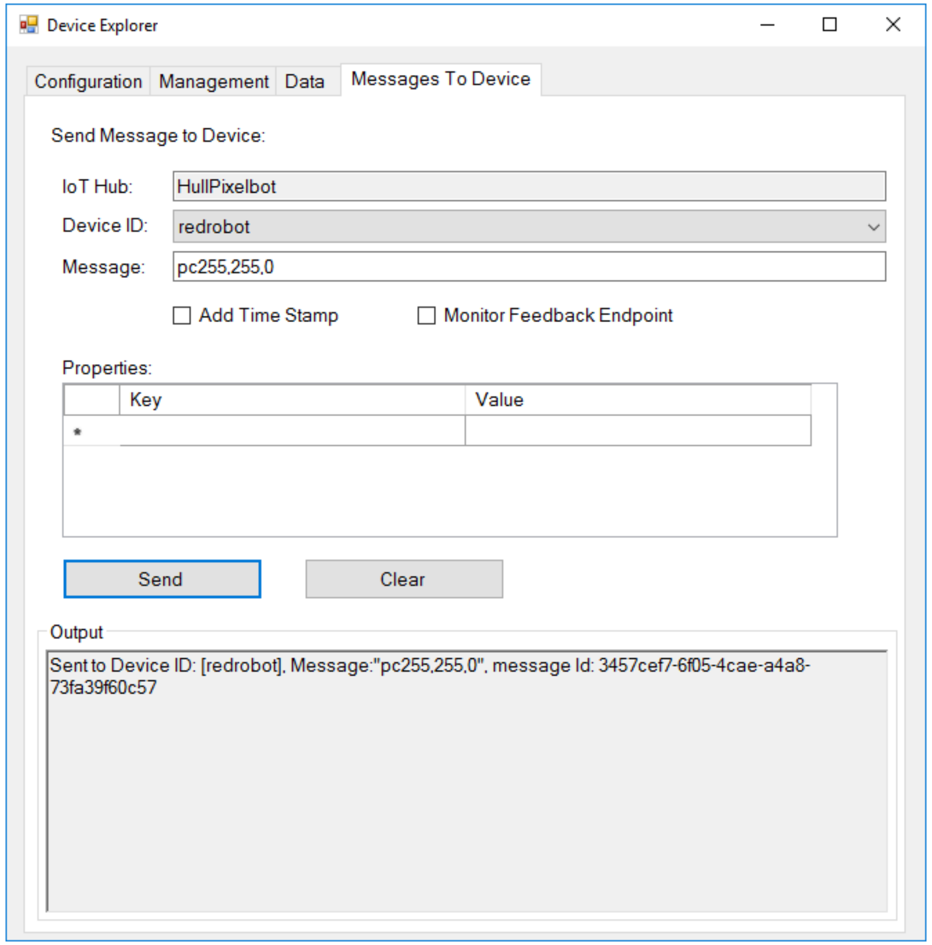
Hostname
HullPixelbot.azure-devices.net
Pricing and scale tier
S1 - Standard
IoT Hub units
1

10/2/2016 UTC
HULLPIXELBOT
0% TOTAL
MESSAGES
14 / 400k
DEVICES
4

Monitoring
HULLPIXELBOT
CONNECTED DEVICES
1
TELEMETRY MESSAGES S...
4

Device Explorer

- The device explorer provides device management and testing
  - This is available in source form
- We can view messages from connected clients and send messages to them as well
- This is not the only way to provision devices
- There is also an api you can use to build a workflow if you have lots of devices



IoT Hub: HullPixelbot
Device ID: redrobot
Message: pc255,255,0

```
Sent to Device ID: [redrobot], Message:"pc255,255,0", message Id: 3457cef7-6f05-4cae-a4a8-
73fa39f60c57
```

Sending an MQTT message from Azure

```
ServiceClient serviceClient = ServiceClient.CreateFromConnectionString(iotHubConnectionString);

var serviceMessage = new Microsoft.Azure.Devices.Message(Encoding.ASCII.GetBytes(message));
serviceMessage.Ack = DeliveryAcknowledgement.Full;
serviceMessage.MessageId = Guid.NewGuid().ToString();

await serviceClient.SendAsync(MQTTName, serviceMessage);

await serviceClient.CloseAsync();
```

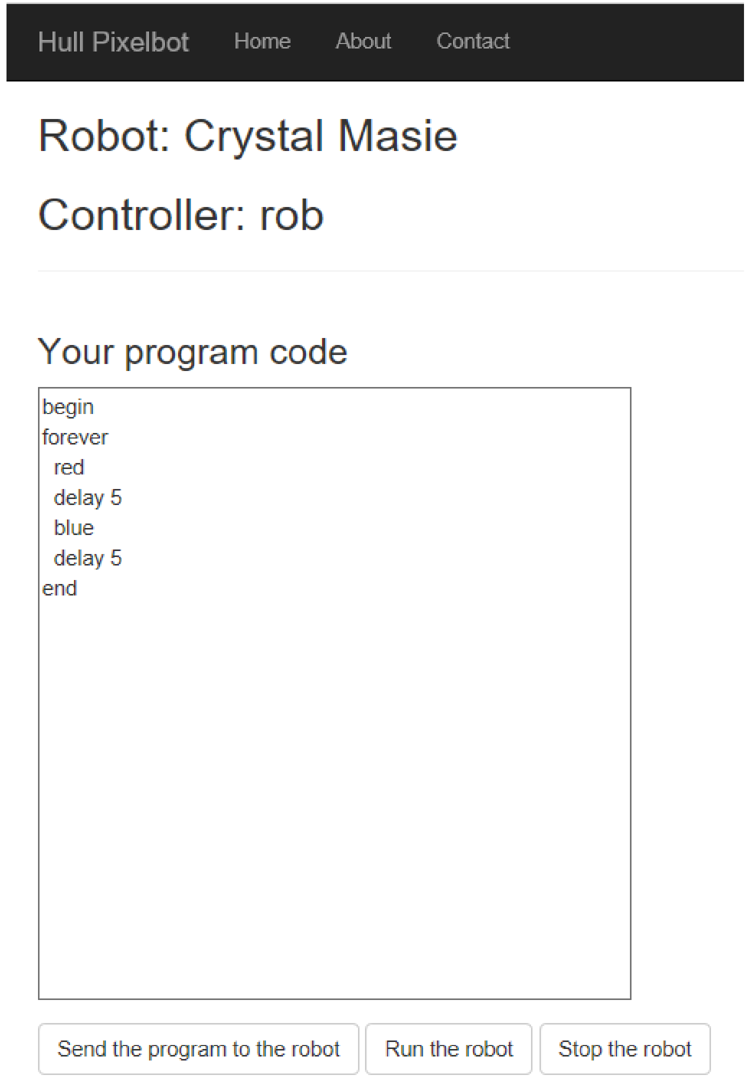
- This is the code that sends a message to the robot from Azure
- Note that we convert the message into ASCII bytes before sending
- When this runs the callback function in the robot runs and picks up the message string – which is a HullOS program

Demo 3: Sending messages to the robot

Creating a web based HullOS code editor

A web based code editor

- Now that we can send messages over MQTT we can start to send programs to our robot
- It might be fun to have a web based program editor
  - You're assigned a robot and can create and deploy programs to it
- So I built one of those next



Robot: Crystal Masie

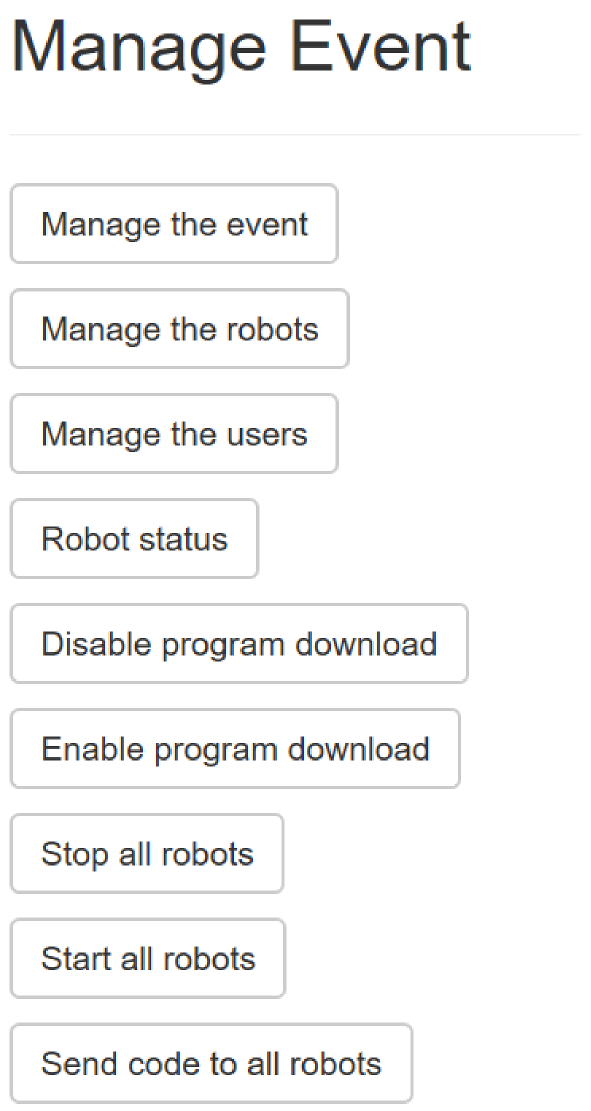
Controller: rob

Your program code

```
begin
forever
  red
  delay 5
  blue
  delay 5
end
```

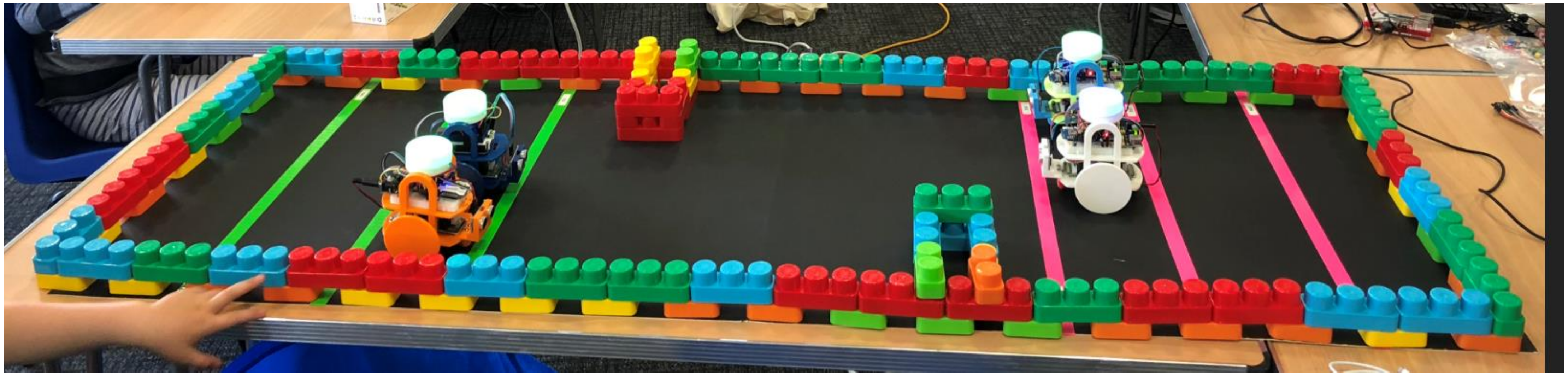
The HullOS code editor

- I’m not very good at ASP.NET applications
- However, it does demonstrate the principles quite well
- The code editor could do with a lot of attention, I would like automatic code keyword completion and better support for mobile code editing



Demo 4: Editing HullOS programs

ROBOT RUMBLE



- The Robot Rumble is a team game that I'm working on
- Teams spend 1 minute programming and 30 seconds running their programs to try and get their robots as far down the field as possible
- I want to get a bunch of people together to do this……

The Hull Pixelbot project

- If you fancy getting involved, or looking at the code and designs you can find out all about the Hull Pixelbot here:

hullpixelbot.com