

# 441190 Extra-Curricular Activities Rather Useful Seminar

## Robots, Names in Lights and Connected Little Boxes

**Rob Miles**

Hull inventor, author, Microsoft MVP and founder of Rather Useful Seminars Rob Miles revels in the joy of making things and connecting them together using Python, JavaScript, C++ and anything else that works in a device that costs five pounds.

Rob will be demonstrating some things that he's made and showing you how to get started in embedded development and have the same kind of fun that he is. And he might take the odd photograph..



**Wednesday 8<sup>th</sup> November**  
**1pm-2pm Wilberforce Lecture Theatre 1**

# About Rob:

- Taught Computer Science at Hull University
  - Spent many years twisting minds and crushing dreams
  - Now back in the university for a guest appearance...
- A Microsoft MVP
- Blogs at: [www.robmiles.com](http://www.robmiles.com)
- Tweets at: @robmiles
- Writes stuff....



Look inside



### Begin to Code with C# Paperback – 9 Sep 2016

by Rob Miles (Author)

★★★★★ 1 customer review

See all formats and editions

Look inside



### Begin to Code with Python Paperback – 8 Dec 2017

by Rob Miles (Author)

★★★★☆ 6 reviews from Amazon.com

Look inside

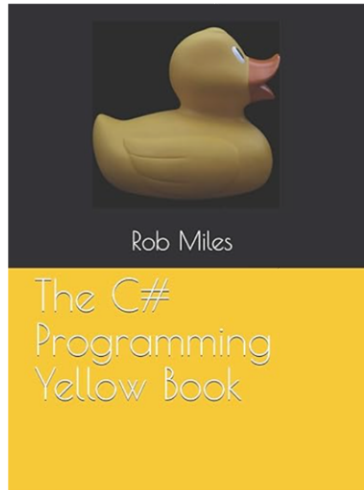


### Begin to Code with JavaScript Kindle Edition

by Rob Miles (Author) | Format: Kindle Edition

★★★★★ 1 review

### Begin to Code: Building apps and games in the Cloud



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



by Rob Miles (Author)

4.5 ★★★★★ 483 ratings

See all formats and editions

Kindle Edition  
£0.99

Paperback  
£9.00 ✓prime

Read with our free app

1 Used from £8.10  
2 New from £9.00

Save 5% on any 4 qualifying items | Terms

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.

Print length

Language

Publication date

Dimensions

ISBN-10



222 pages

English

19 Oct. 2018

21.59 x 1.27 x

1728724961

# HackSpace

- HackSpace magazine is a fantastic read

- And you can read it for free 😊

[hackspace.raspberrypi.com/issues](https://hackspace.raspberrypi.com/issues)

Make a Pico MIDI 'crackers' controller and a Pure Data sound synthesizer  
TUTORIAL

## Make a Pico MIDI 'crackers' controller and a Pure Data sound synthesizer

Create synth sounds and hardware to control them

**Rob Miles**  
@robmiles

Rob Miles has been playing with hardware and software since almost before there was hardware and software. You can find out more about his so-called life at [robmiles.com](https://robmiles.com).

**THE PICO MIDI CONTROLLER**

The article 'Make a Pico musical cheese box', in HackSpace issue 48, described how to use a Raspberry Pi Pico to make a device that transmits MIDI notes and drum patterns. We used it to control a Pure Data music patch running on a Raspberry Pi, which made sounds from audio samples and sine waves. In this article, we are going to create a 'crackers' controller to go with the cheese box. Then

we'll make a Pure Data digital synthesizer to run on the Raspberry Pi to create interesting sounds, and use the crackers controller with it.

**INSIDE THE BOX**

Figure 2 shows the contents of the box. The wiring looks a bit crackers (hence the name), but it worked first time. The trick was to have a full circuit diagram before starting construction, and then connecting each cable in turn.

Figure 3 shows the circuit for the box. Two ground connections are used: one for the pixel rings, and another for the buttons and rotary encoders. You could control more values by adding more buttons and pixel rings.

**Figure 1**

The 'crackers' MIDI controller is on the left. The red parts of the rings around each of the four knobs indicate the level of that setting. The other colours (blue, green, yellow, and magenta) indicate which setting is being controlled by that encoder. The musical cheese box is on the right.

# Agenda

- A look at a few devices
  - Pomodoro Timer
  - Bluetooth Chord keyboard
  - MIDI CheeseBox and Chocolate SynthBox
  - Hull Pixelbot
  - Lights in Names/Connected Little Boxes
- Making a product
- The joy of making things

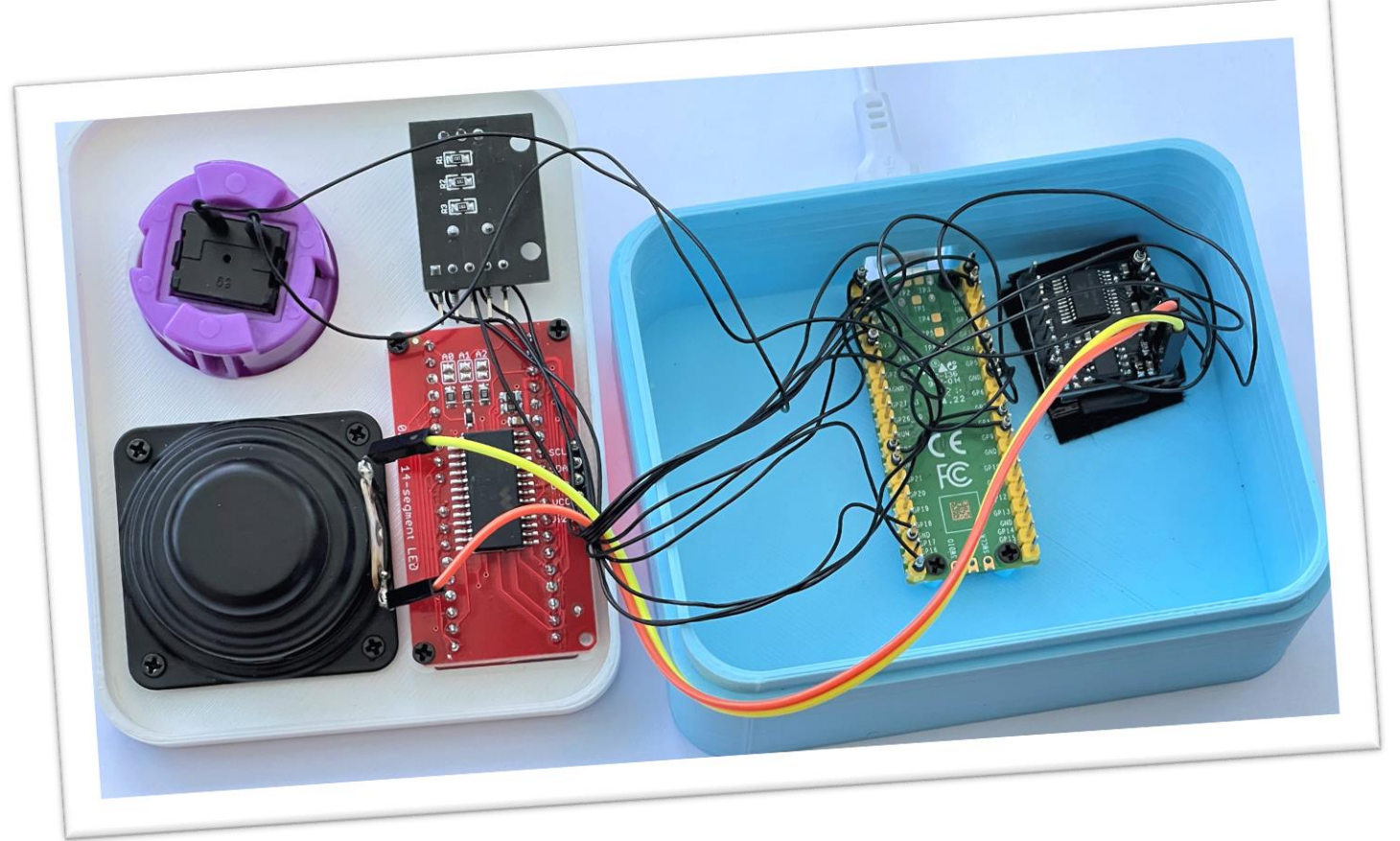
# Pomodoro Timer

# Pomodoro Technique



- The Pomodoro Technique aims to improve productivity by partitioning work into a series of units
  - It was described in the book "The Pomodoro Technique" by Francesco Cirillo
  - It was named after the tomato shaped timer used by the author
- I wanted a digital one that would talk to me
- So, I made one

# Timer Internals



- I used a Raspberry Pi PICO, digital music player, rotary encoder and text display along with a large purple button
- The box is programmed in Circuit Python



# Timer Case

- The case was designed with FreeCAD
- The design was generated by a Python program running inside the design tool
- This lets you place the components and then builds a box around them
- I use this for most of my project builds:
  - “3,000 lines of technical debt”



# Bluetooth Chord Keyboard

# MicroWriter Agenda

- Many years ago I had a pocket organizer which had a chord keyboard



- Text could be entered by entering chords on the device
  - Although they kept an alphabetic keyboard for marketing reasons
- I used this a lot, and wrote quite a bit of code for it

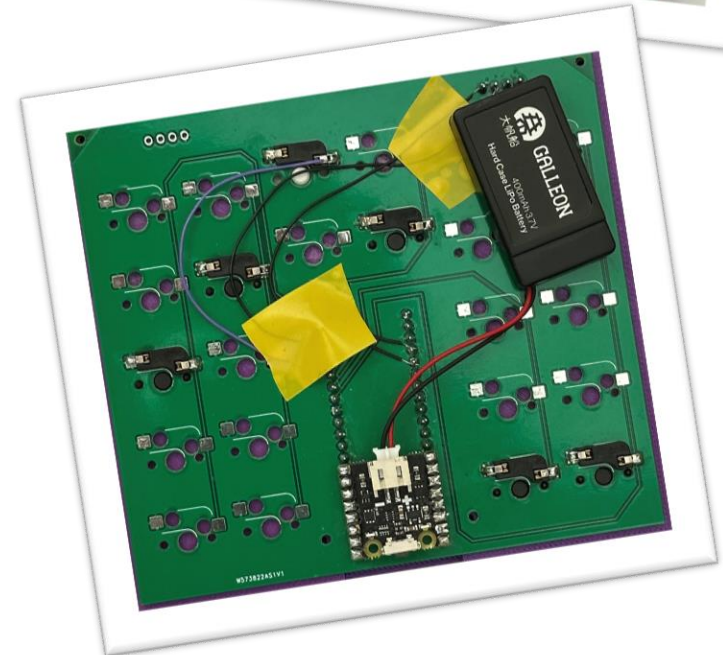
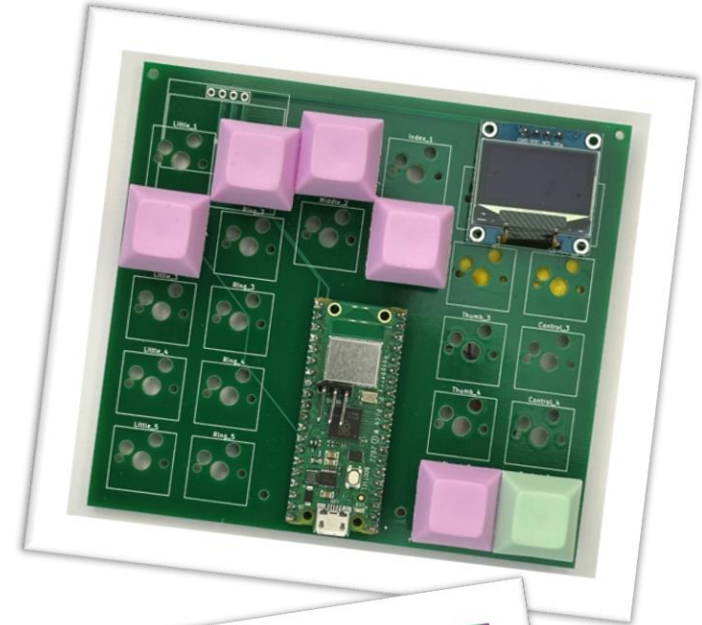
# Raspberry Pi Powered keyboard



- I thought I'd create a left and right-handed chord keyboards
- These are battery powered and connect via Bluetooth

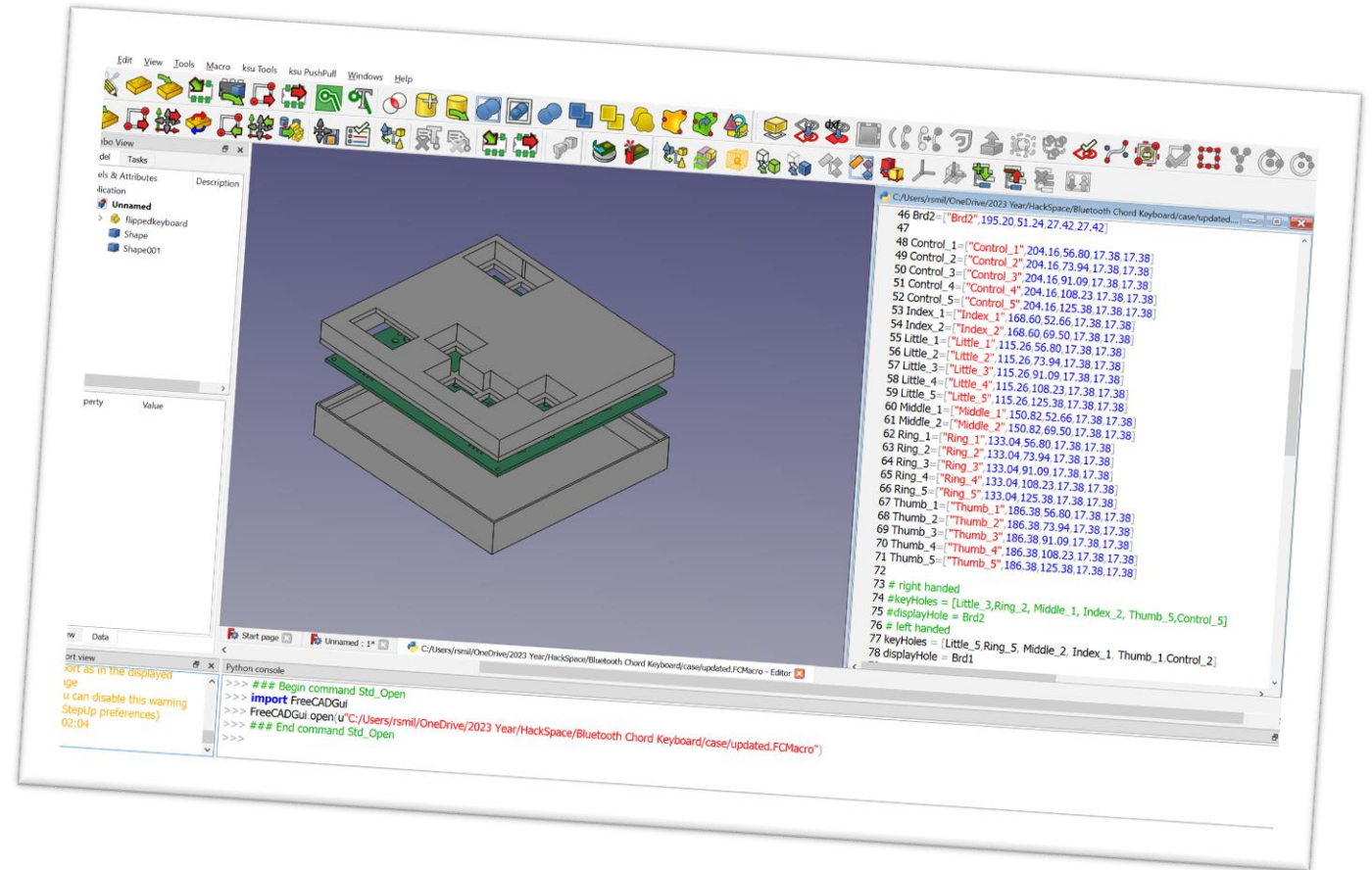
# Custom PCB

- I designed a custom PCB and had it made in China
  - This was surprisingly cheap to do
- I only made two mistakes...
- I think everyone should design at least one PCB in their life
  - You can etch images and all kinds of fun stuff on them
- The KiCAD program is free and worth learning how to use



# Integrated design

- I ended up running a Python program inside KiCAD to extract the component placement information I needed to make the box in FreeCAD



- There is even a KiCAD plugin for FreeCAD so that you can design your hardware and PCBs in the same tool
- Great fun

Chocolate Synthbox, Crackers  
Controller and MIDI CheeseBox

# Music

- This is a collection of devices that can make a kind of music
- The controllers talk MIDI to a Raspberry Pi

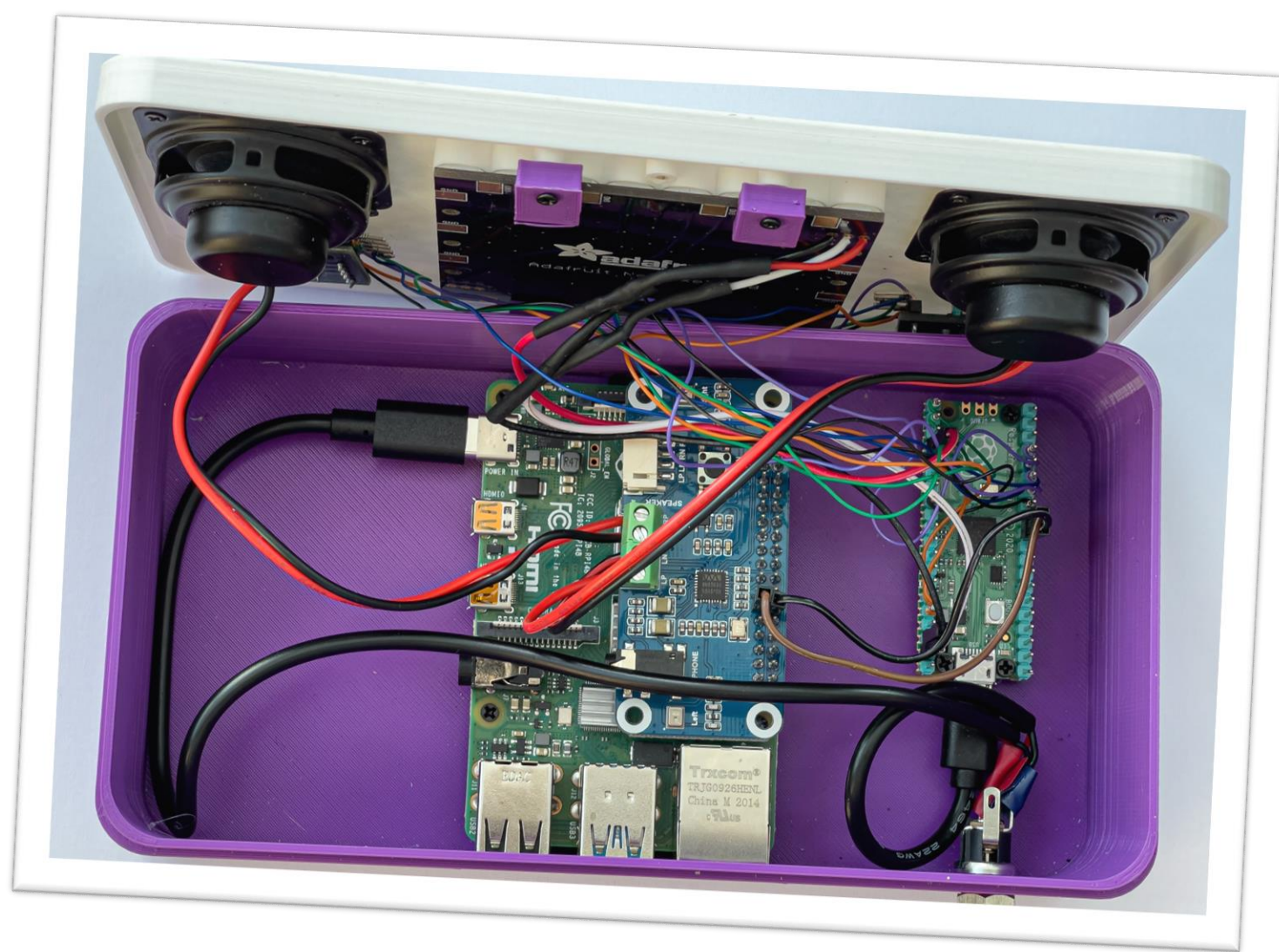


- The sound is produced by programs written in “Pure Data” which is a graphical language created for making sounds
- The controller generates MIDI control messages
- The “Cheesebox” plays notes and works as a sequencer



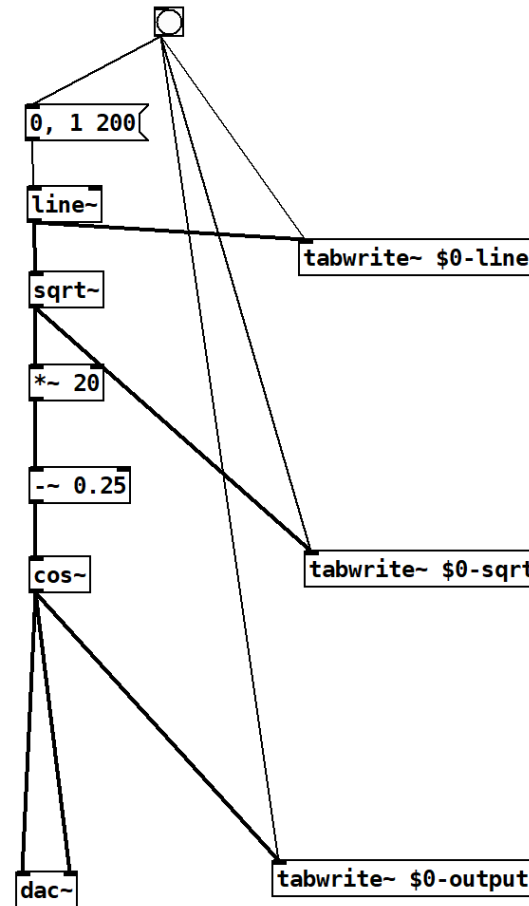
# Hardware

- The music box uses a Raspberry Pi to make the audio and a PICO to drive the graphical display and read the encoders
- The sound is produced by an audio hat which also contains an amplifier



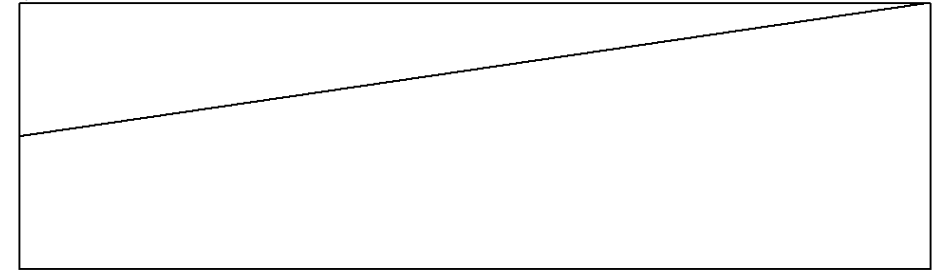
# Pure Data

- Pure Data can draw graphs of the audio signals
- You can see that the line becomes a curve when we take the square root of the values
- Which makes a more interesting sound wave when we use it to drive a sine waveform



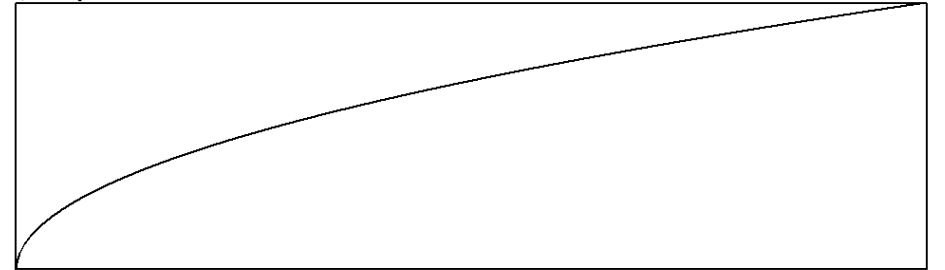
Linear ramp that goes from 0 to 1 in 200 milliseconds

\$0-line



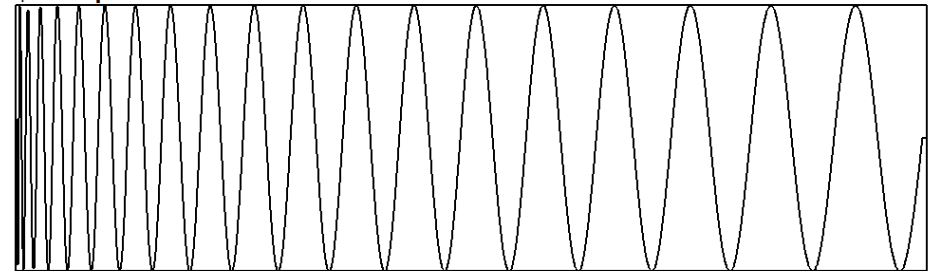
Shaped Ramp which is steep at the start and flattens out

\$0-sqrt



Output signal of a shaped sine wave

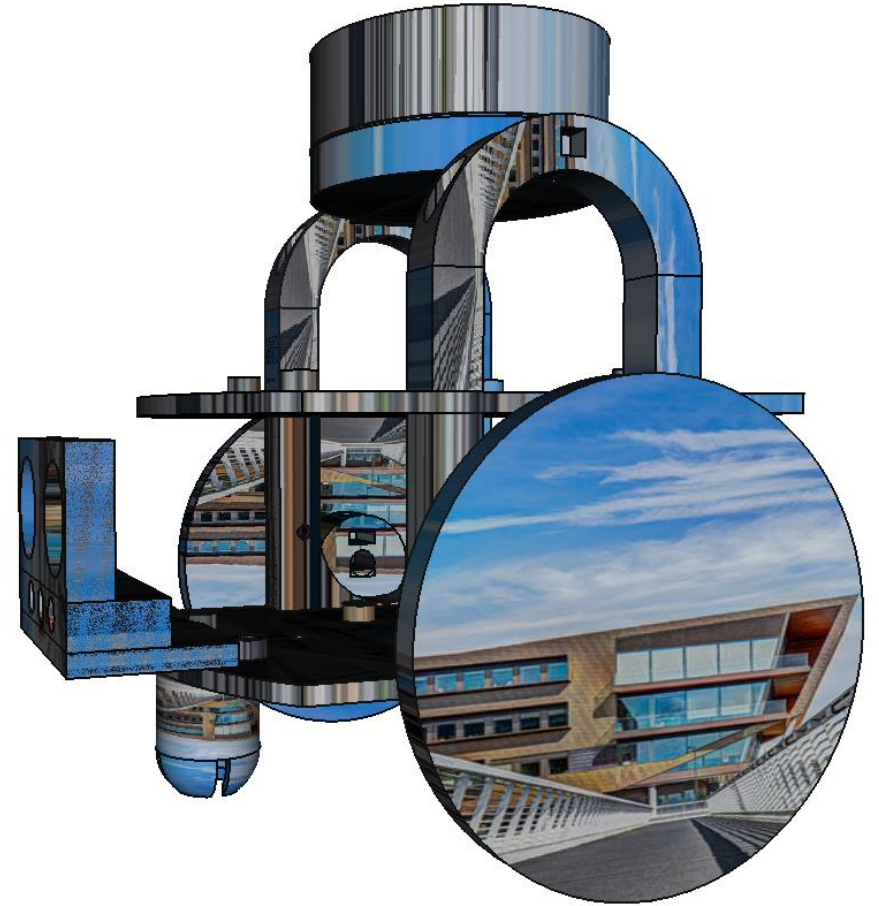
\$0-output



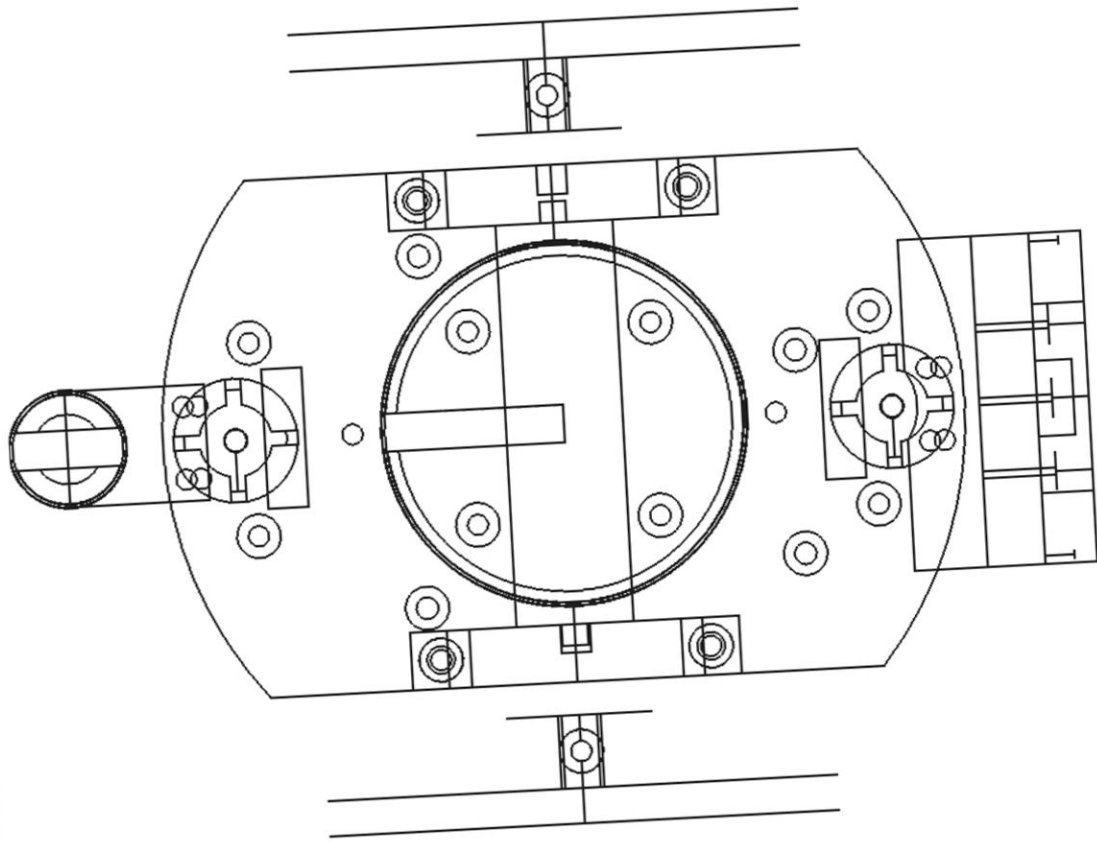
Hull Pixelbot

# 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



# Open Source

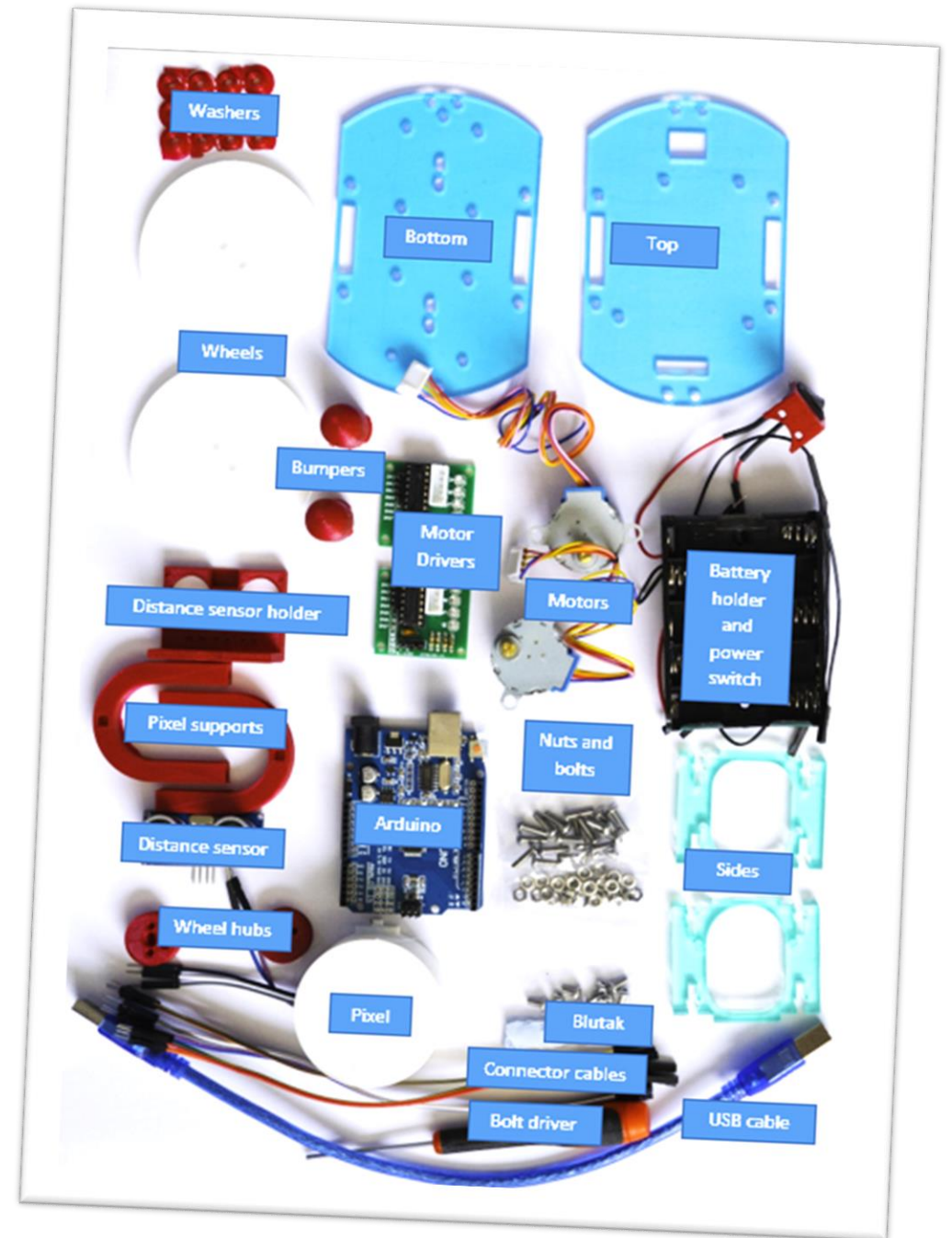


- The designs for the robot and controlling software are all open source
- Everything is published on GitHub

<https://github.com/HullPixelbot>

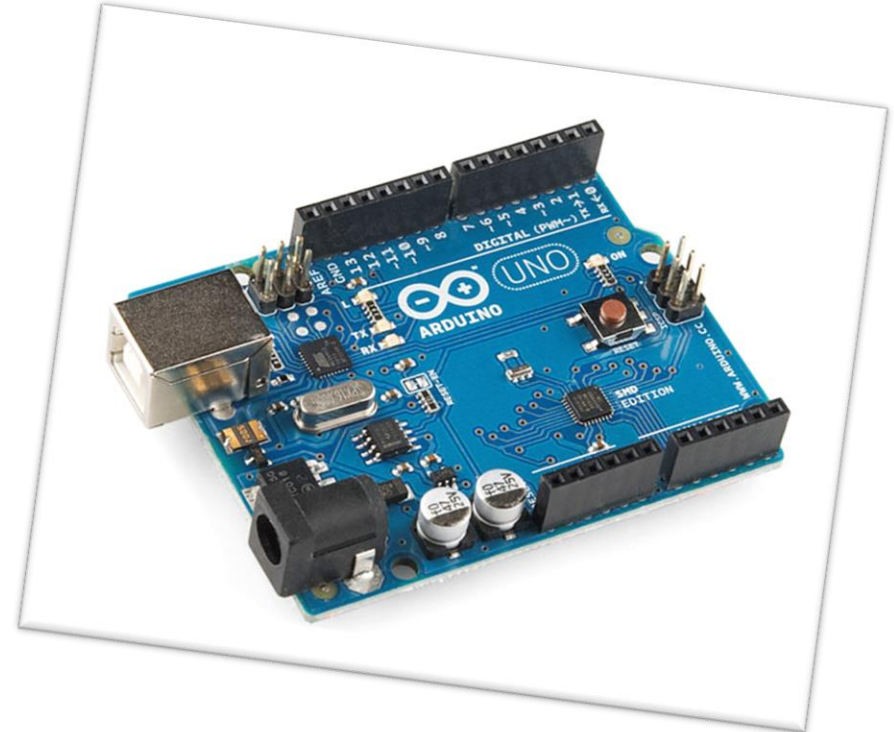
# Fun to build

- 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



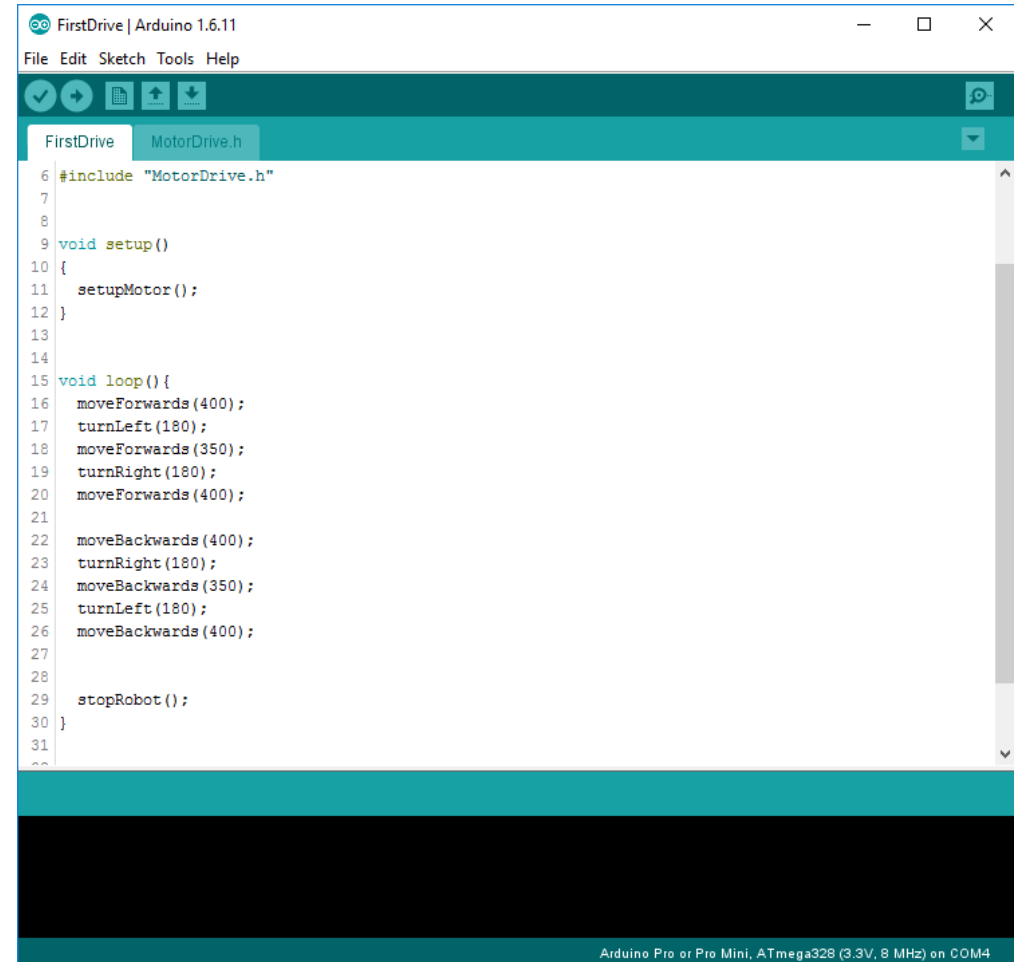
# The Arduino Uno

- 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



The screenshot shows the Arduino IDE interface with a sketch named 'MotorDrive.h'. The code is as follows:

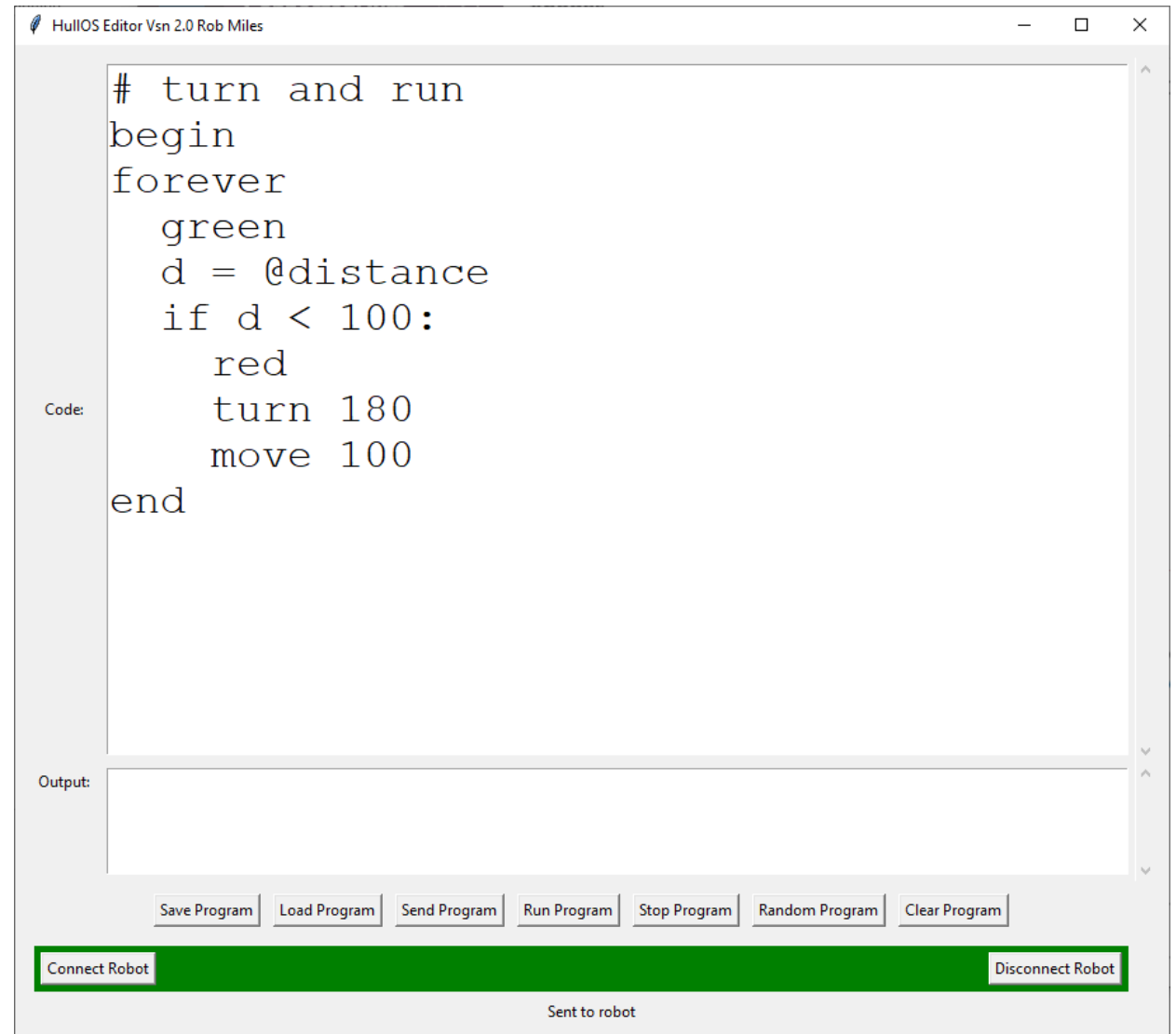
```
6 #include "MotorDrive.h"
7
8
9 void setup()
10 {
11   setupMotor();
12 }
13
14
15 void loop() {
16   moveForwards(400);
17   turnLeft(180);
18   moveForwards(350);
19   turnRight(180);
20   moveForwards(400);
21
22   moveBackwards(400);
23   turnRight(180);
24   moveBackwards(350);
25   turnLeft(180);
26   moveBackwards(400);
27
28
29   stopRobot();
30 }
31
32
```

At the bottom of the IDE, the hardware configuration is shown: 'Arduino Pro or Pro Mini, ATmega328 (3.3V, 8 MHz) on COM4'.



# Easy to program

- HullIOS 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 time/sliced platform



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

Code:

Output:

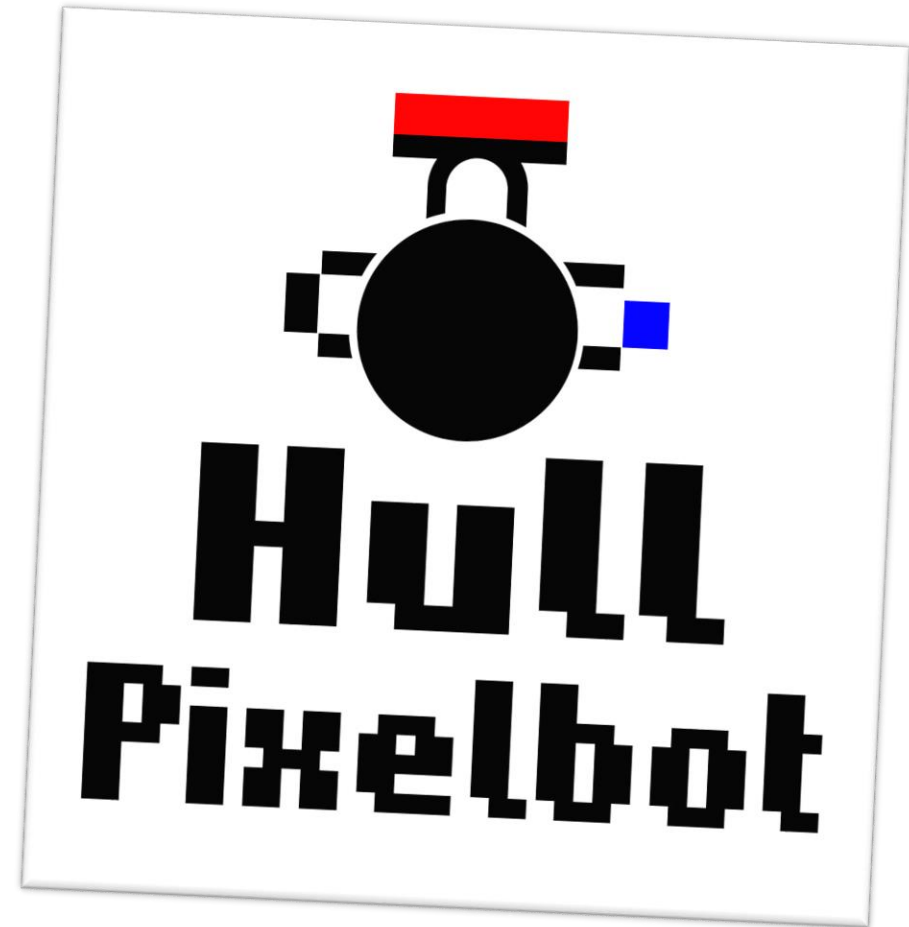
Save Program Load Program Send Program Run Program Stop Program Random Program Clear Program

Connect Robot Disconnect Robot

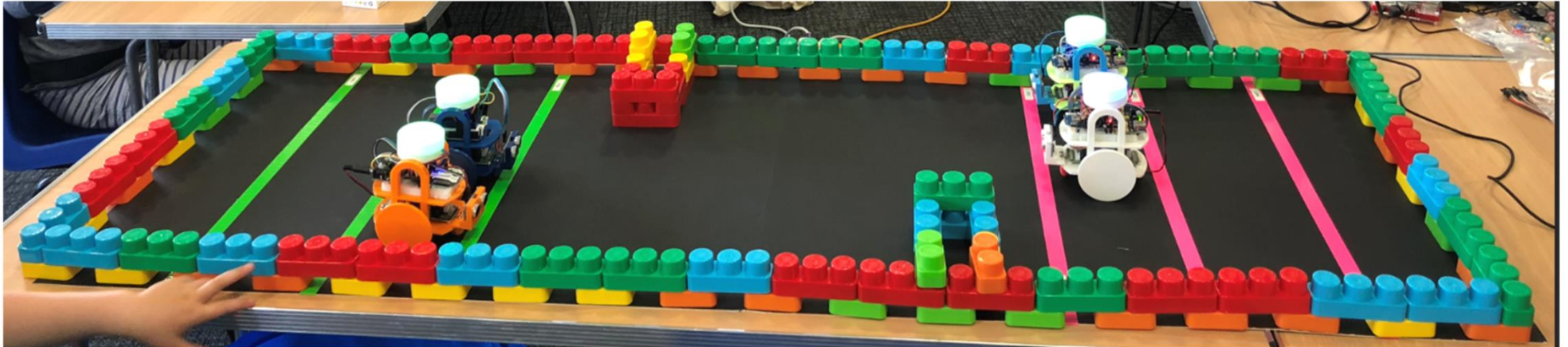
Sent to robot

# Remote control

- I've added an ESP8266 device on top of each robot
- This handles Wi-Fi connection and lets us control the robot remotely
- I've got a web backend which lets robot controllers create programs and run them on the robot



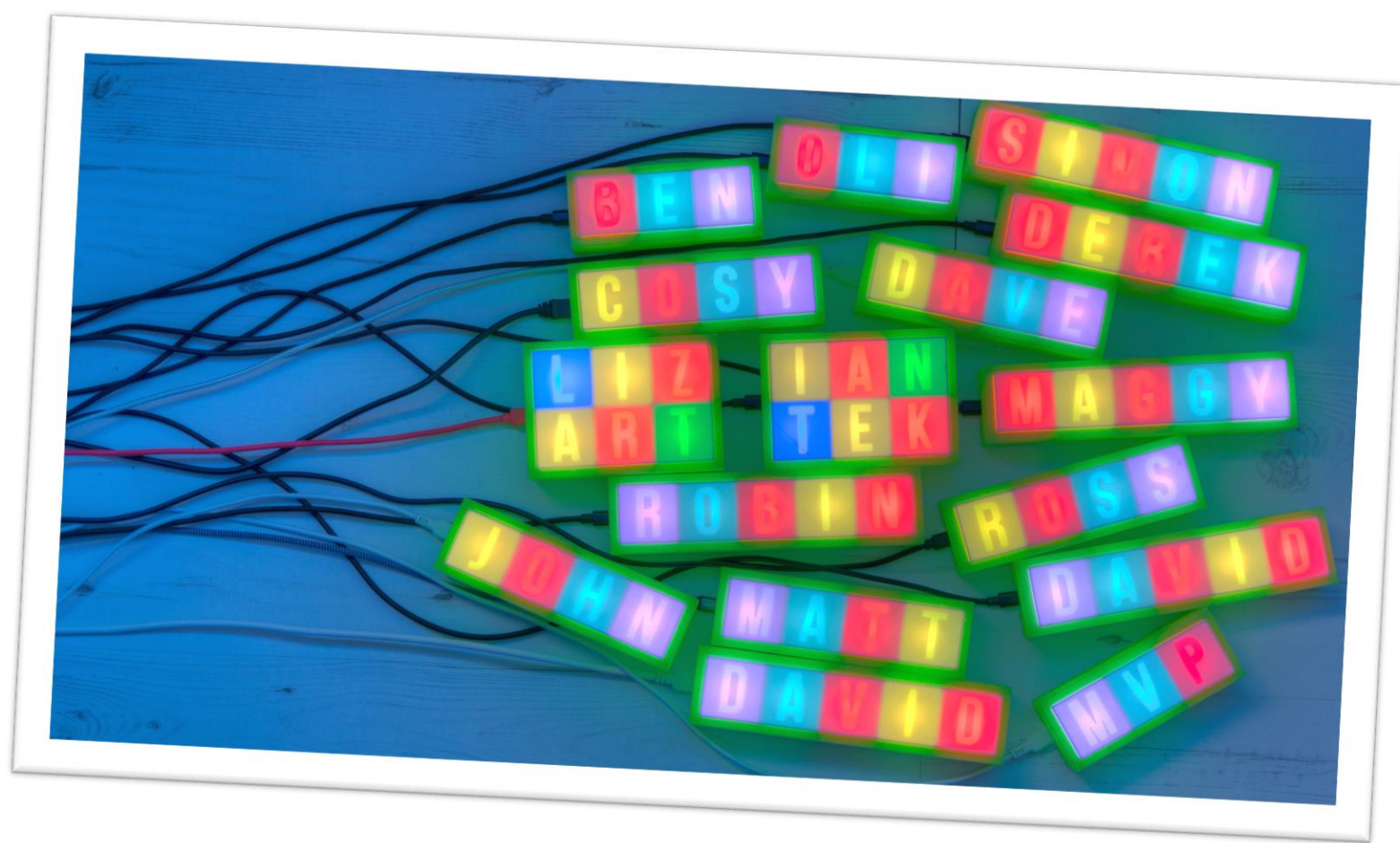
# Robot Rugby



- The Robot Rugby 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.....

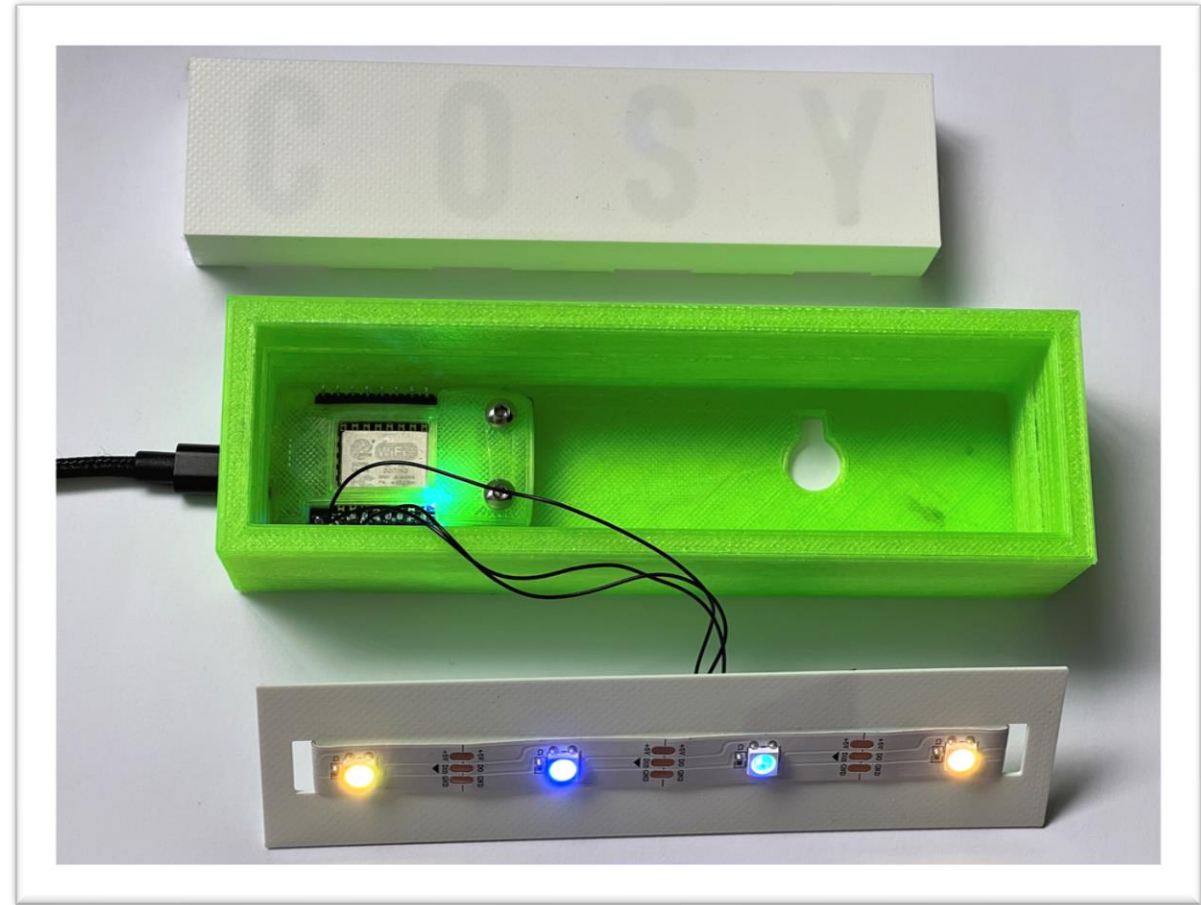
Lights in Names/  
Connected Little Boxes

# Lights in Names



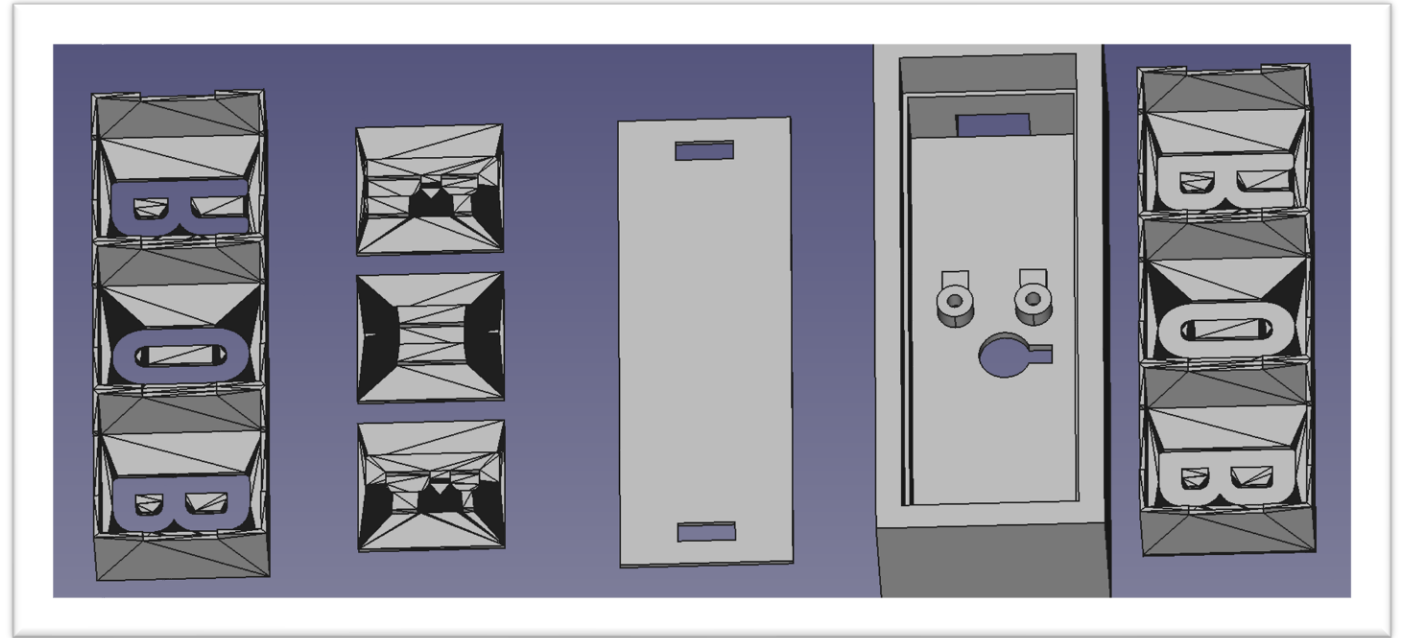
- This started as a vanity project but people seemed to like them
  - Perhaps everyone is a bit vain..
- Each light contains a connected microcontroller (ESP8266) and some neopixels

# Put your name in lights



- The hardware is very simple to make
- It doesn't need to be connected

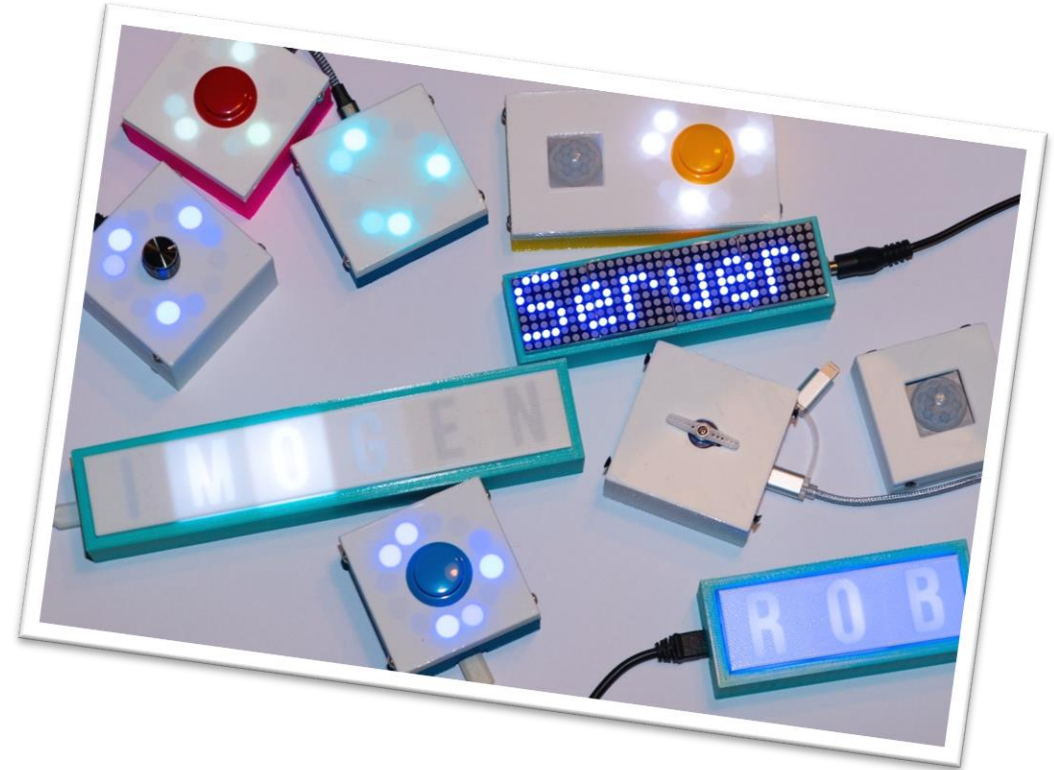
# Light Boxes



- Another Python program runs within FreeCAD to make custom STL files to print for each light
- You could use this to manufacture customised lights
- A light is a form of connected little box....

# Connected Little Box

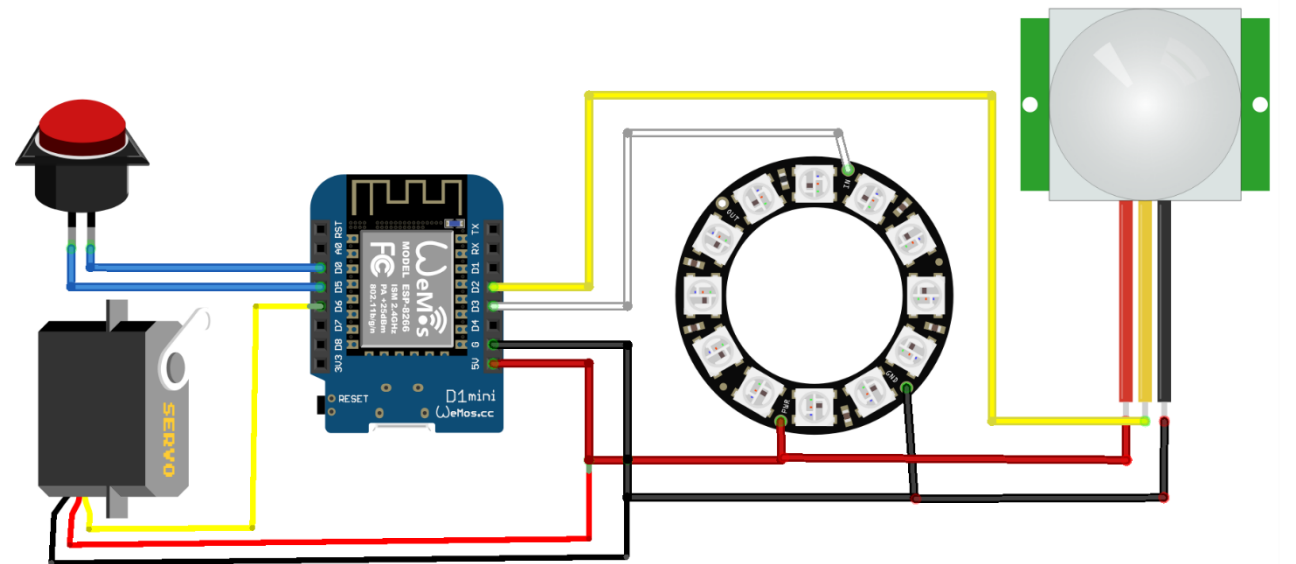
- A connected little box contains a WiFi enabled microcontroller and connects to sensors and outputs
- All boxes run the same software which is configured for that specific box
- Configuration can be performed locally via the serial port or remotely over a network connection
- There are default hardware configurations for a range of devices





# Example configuration

- This box has a button, a servo, a PIR sensor and a pixel ring
- All these peripherals can be connected and controlled simultaneously
  - The PIR or button inputs could be used to trigger the servo and/or the pixel ring
- A box can also send messages to control peripherals on other boxes



# Internal software

- The software is implemented as a number of *processes* and *sensors*
- A process runs to control something
- A sensor runs to fetch data from a source
  - Each implements a state machine
  - Each has a collection of setting values
  - Each is initialised when the box starts and then repeatedly updated when the box is active
  - Each should perform a non-blocking action
- It is easy to add new processes and sensors

# Connecting Little Boxes

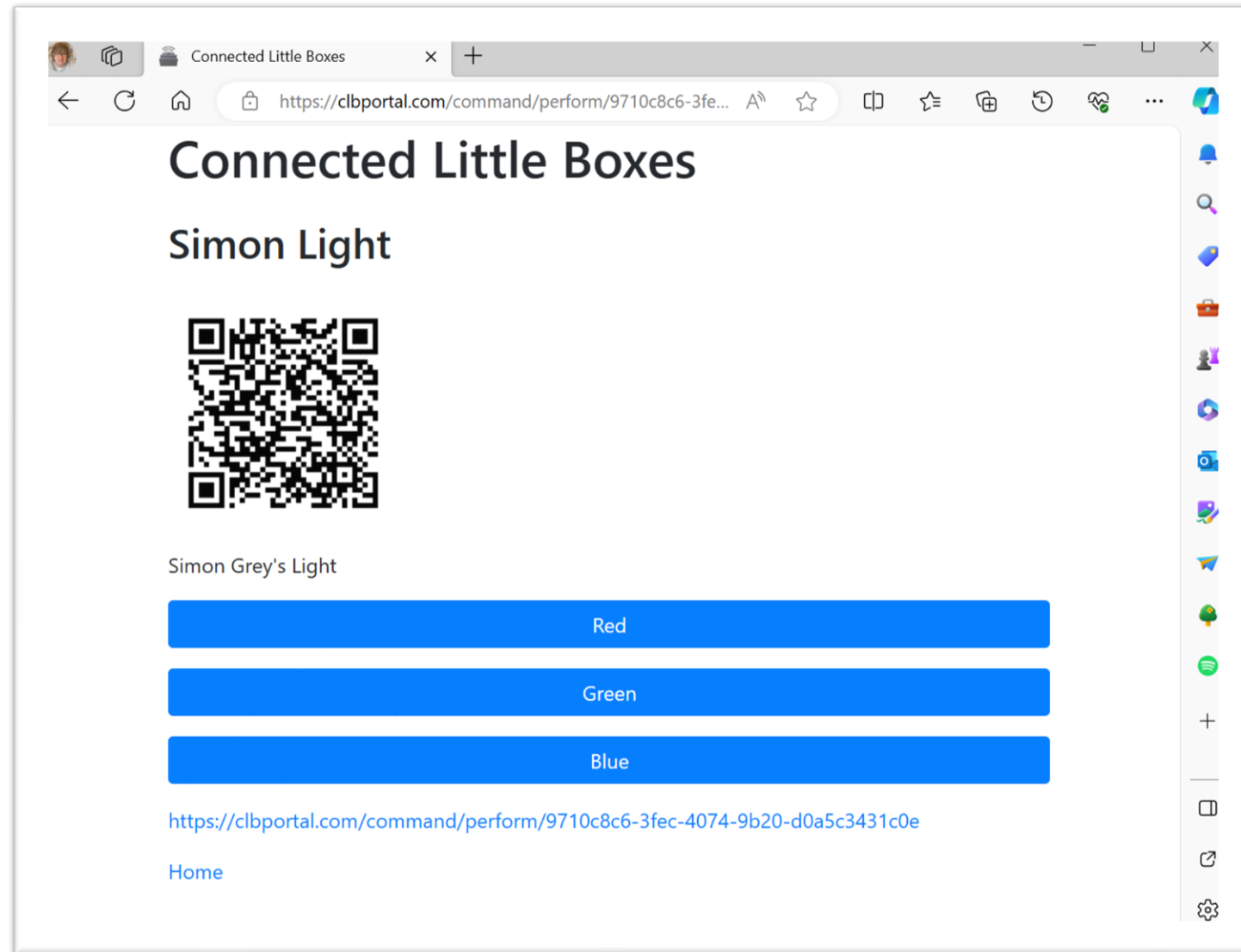
- The devices that we have looked at all have WiFi
- You can use them to create network connections so they can use datagrams or connections
  - They will work as web servers
  - They can also support secure sockets
- You can connect to network services using restful connections
- However, the IoT community makes a lot of use of MQTT (Message Queue Telemetry Transport)
- This is a very easy way to hook sensors and actuators together

# 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
  - Although setting up your own broker is easy enough
- It also runs (surprise surprise) on the esp8266, esp32 and PICO
- It is a great way to create cheap, connected, sensors

# Web Backend

- The web backend lets you create command groups containing commands to be selected and sent to a box
- These can be accessed from anywhere



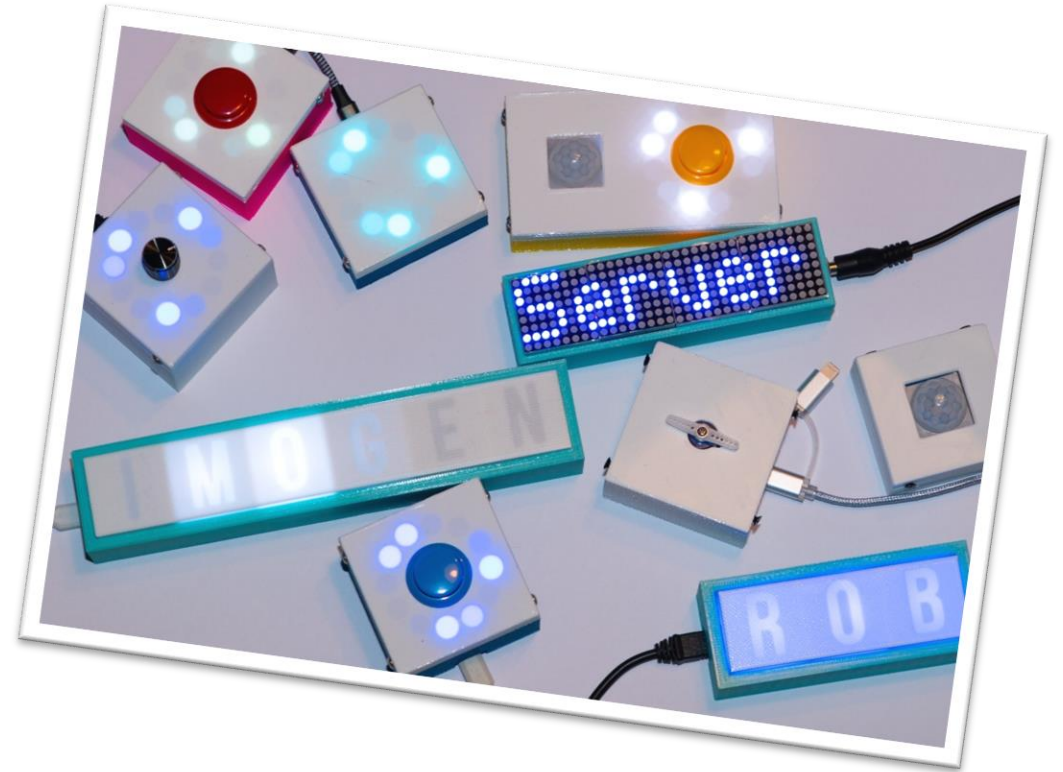
# Try it!

- Scan the text and see if you can change the colour of the Simon's light



# Connected Little Boxes

- Boxes can send commands to other boxes and events (for example turning a control or pressing a button) can trigger commands
- The box can persist commands and run them on sensor and time triggers
  - Press a button to open a door
  - Display the temperature every minute



# Other things...

- The devices that you have seen are just a few of the ones that I've made over the years
  - Trombone controller
  - Party camera
  - Furby remote control
  - Web controlled flashlight
  - Bluetooth controlled flashlight
  - 3D printed camera
- You can find them all in back issues of HackSpace magazine
- You should be thinking about making your own things...



# The Joy of Making Things

# You should make things

- There has never been a better time make interesting devices
- You can go from idea to device in a very short time
  - The toolchains are easy to use
  - The devices are cheap to buy
  - They are easy to connect



# Toolchains

- You can write in C, Python, JavaScript or even C#
  - Although I'd use this as an excuse to play with a language you've not used before
- You can use Visual Studio Code and PlatformIO to build for these and even debug code running inside the device
- The tools are all free to use and work on PC, Mac or Linux
- ChatGPT knows a lot about embedded development 😊
  - Although quite a bit of it is wrong 😞

# Devices

- You can get an internet connected device you can program in Python for around a fiver
  - Take a look at the ESP32 and Raspberry Pi PICO-W
  - These can even host Wi-Fi and websites
- If you spend a bit more, you can get a Raspberry Pi Zero
  - This will run JavaScript under nodejs and gives you some serious power and networking capabilities
- You can get all kinds of interfaces, servos, motors and displays for pocket money prices – as long as you are prepared to wait for them to arrive from China

# Connection

- Not all devices need a network connection
  - The Pomodoro timer just sits and works
- However, making things connected makes them much more interesting
  - Devices can connect to your home network, or a phone hotspot
  - Protocols like MQTT allow simple inter-device communication
  - Devices can host websites, databases and do all kinds of networked things
  - They can even use HTTPS for secure communications

Getting Real

# Making a product

- Getting something to work is actually the easiest bit
- Once you've done that you have three more things to worry about if you want to make something of your idea:
  - Security
  - Privacy
  - Quality
- Some of these require a totally different skill set from hardware and software development
  - You need to know that these things are important, even if it might not be you that ends up dealing with them

# Security

- You can start with simple, open, protocols but you need to be aware that these are vulnerable
- You need to consider how worried you (and your customers) are about that
  - Nobody would care much if someone took over my PixelBot, but they might not like it if someone stole their Wi-Fi credentials from their robot
- Modern devices support secure storage and certificates



# Privacy

- An embedded device can learn a lot about its user
- A connected embedded device can share this information
- You need to be mindful of your obligations when you make a device available for others to use

# Quality

- It doesn't matter if something that you have built for fun goes wrong
  - In fact, sometimes it makes things more interesting
- However, once other people start to use things you've made for them, the situation changes
  - If my Pomodoro timer goes wrong and someone misses an important meeting because of it, I might be liable to compensate them
- Quality also extends to other parts of a product:
  - Documentation
  - Support



# Don't Panic!

- I'm not saying these things to put you off making things
- I'm really saying that you need to be mindful of these issues
- They are not things that should stop you from taking your ideas to market
- They are just things that it is useful to be aware of up front



Going Forwards

# Adding value to yourself

- One of the reasons that you come to university is to add value to yourself
  - Make yourself better at doing stuff and working with others
- Making embedded devices is a great way to add value while having fun and learning stuff
  - It gives you lots of things to talk about at interviews



# Links:

- C# Yellow Book:  
<http://csharpcourse.com/>
- HackSpace Magazine:  
<https://hackspace.raspberrypi.com/>
- Pomodoro Timer Repository:  
<https://github.com/CrazyRobMiles/PICO-Talking-Pomodoro>
- FreeCAD design tool:  
<https://www.freecad.org/>
- Bluetooth chord keyboard  
<https://github.com/CrazyRobMiles/PICO-blue-chords>
- KiCAD pcb layout tool:  
<https://www.kicad.org/>
- Pure Data audio manipulation language:  
<https://puredata.info/>
- Pico MIDI Cheesebox Repository:  
<https://github.com/CrazyRobMiles/PICO-MIDI-Cheesebox>
- Crackers Controller Repository:  
<https://github.com/CrazyRobMiles/PICO-MIDI-Crackers-Controller>
- Hull Pixelbot Repositories:  
<https://github.com/HullPixelbot>
- Connected Little Boxes Respositories:  
<https://www.connectedlittleboxes.com/>
- Lights in Names Repository:  
<https://github.com/connected-little-boxes/lights-in-names>
- PCB manufacture:  
<https://www.nextpcb.com/>  
<https://www.pcbway.com/>
- Buying parts:  
<https://shop.pimoroni.com/>  
<https://coolcomponents.co.uk/>  
<https://robosavvy.co.uk/>  
<https://www.aliexpress.com/>  
<https://thepihut.com/>