Lazydoro

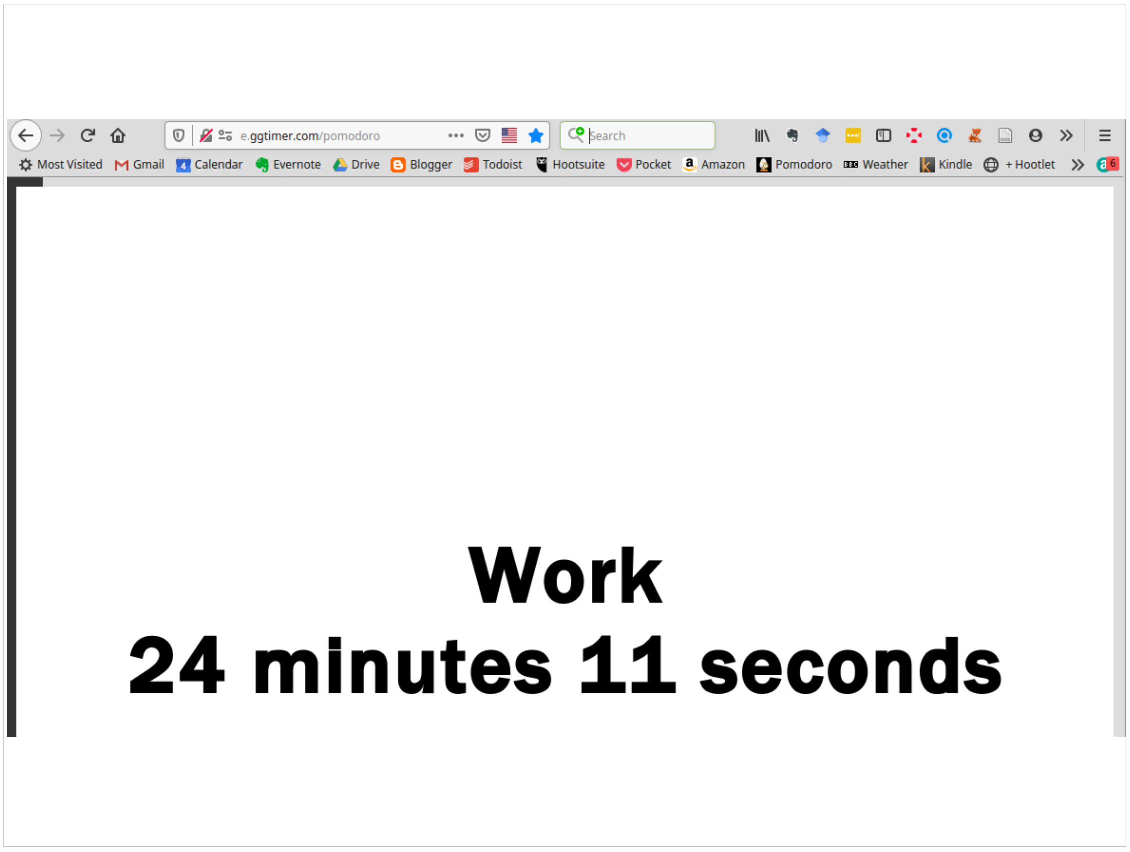
a Pomodoro Timer for the Lazy Developer

Romilly Cocking
@rareblog

The Pomodoro technique is a great way to stay focussed.

Here's how it works:

1) set the timer for 25 minutes.
2) if you stop before then, reset the timer when you restart.
3) When the timer goes off, tally the competed Pomodoro and take a five minute break.
4) After four Pomorodos, take a 25 minute break.



Work
24 minutes 11 seconds

There's a free browser-based timer at e.ggtimer.com.

I've used it for years. It works well **if** you remember to set it going!

Pomodoro works

If you remember to start the timer!

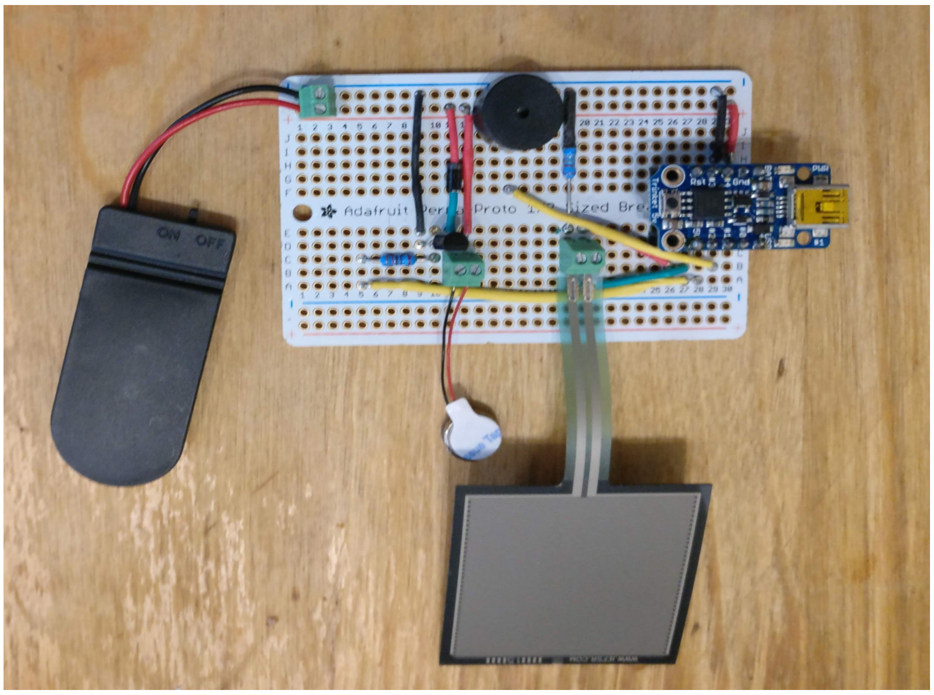
BJ Fogg to the rescue

I decided to use one of BJ Fogg's techniques – change behaviour by changing my environment.

BJ Fogg is a world expert on human behaviour, habit formation and habit change. I took a course of his years ago and it's really worked for me. He's recently published a book which has lots more detail about his techniques.

One way to change your behaviour is to add or remove a prompt (or cue).

I started work on a Pomodoro timer that started automatically whenever I sat down at my desk.

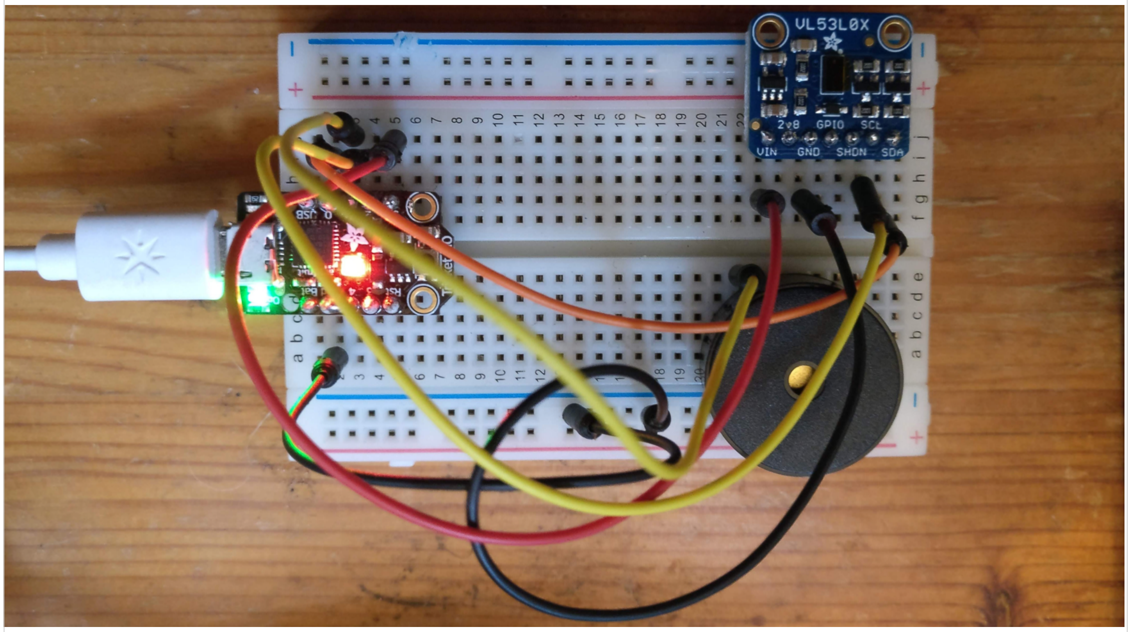


I planned on using a pressure-sensitive resistor inside a cushion. I called the project <cough> Cushadoro.

It worked (sorta) but there were problems.

1) Electronics does not like being squashed
2) The battery ran out without any external indication, and then had to be replaced.

I gave up for a while until Richard Kirby gave me a better idea.

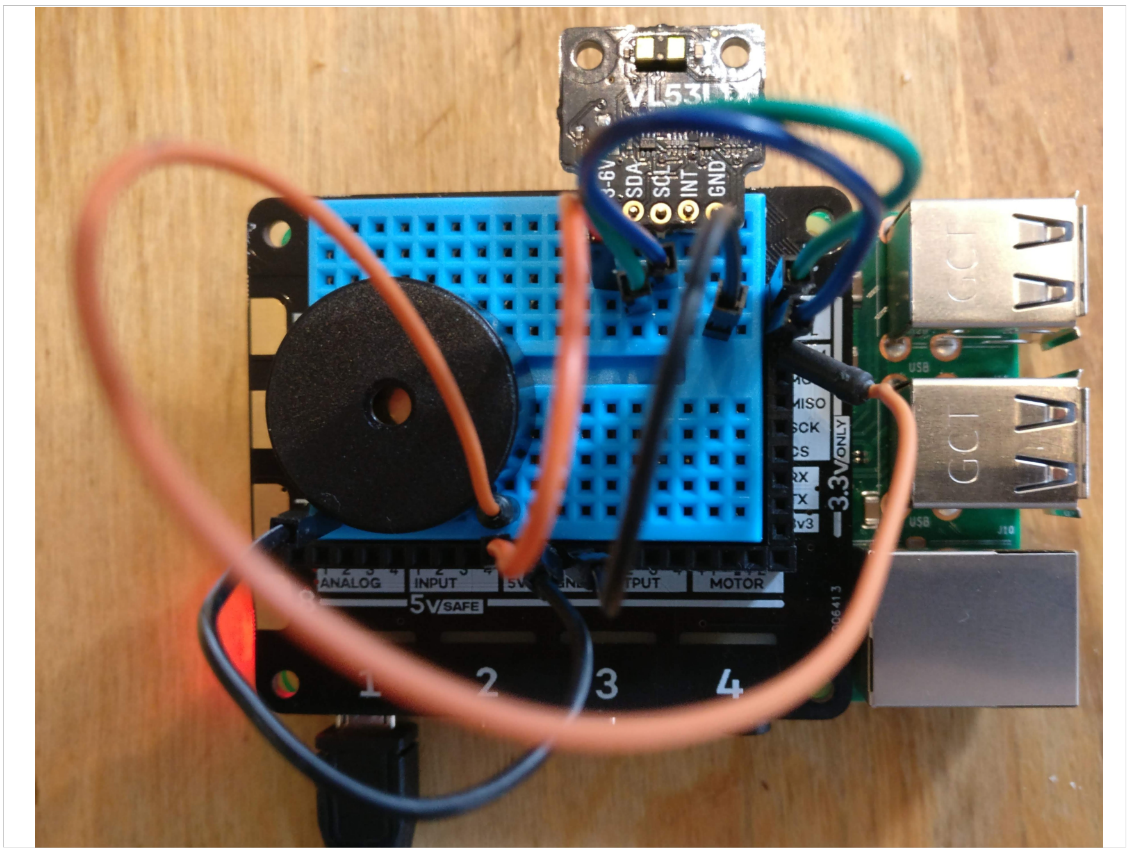


Here's lazydoro Mk 1. It uses a ToF (Time-of-flight) sensor to detect when I am at me desk.

The CPU is an Adafruit Trinket M0, programmed in CircuitPython. When I've been at my desk for 25 minutes it sound a buzzer, and when my 5 minute break is up it sounds a different tone.

If I leave my desk during a Pomorodo the timer restarts.

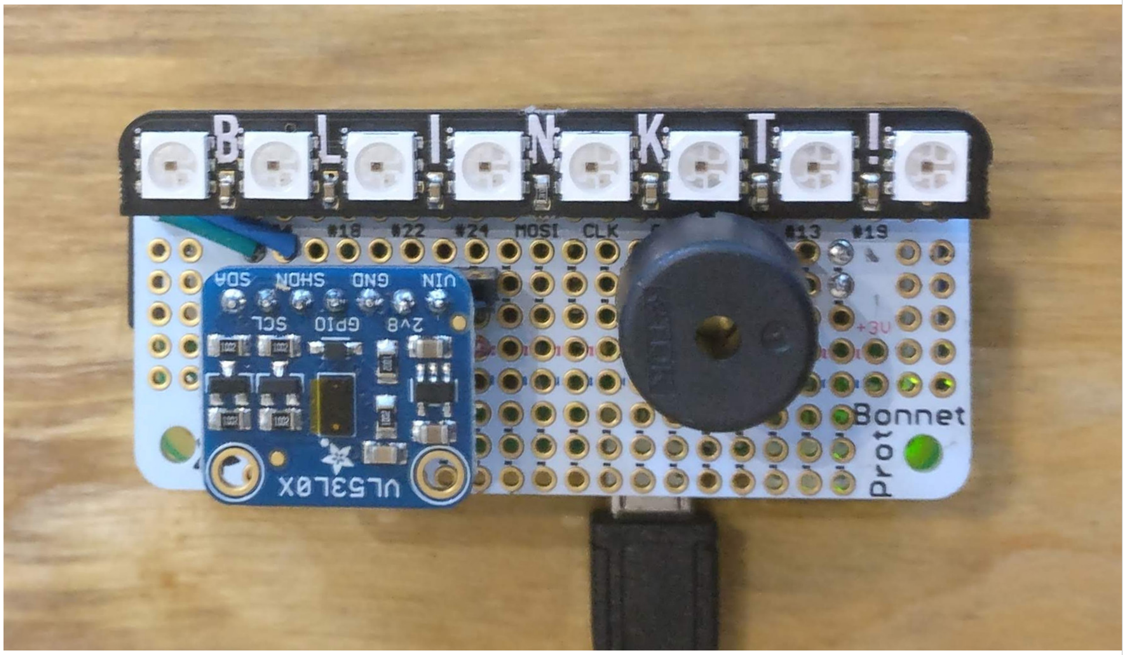
It works pretty well but it doesn't count Pomodoros and it needs to be turned on and off.

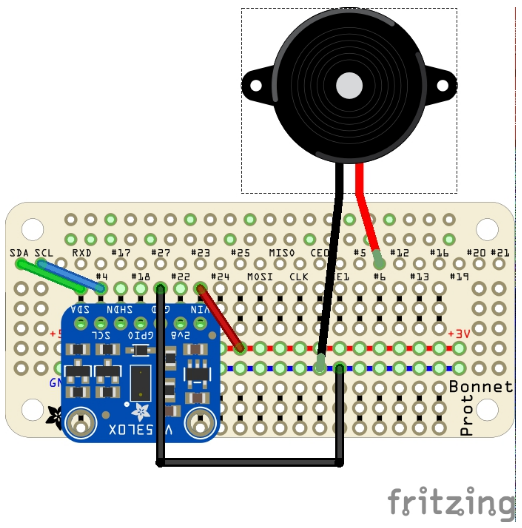


Here's the Mk 2 prototype.

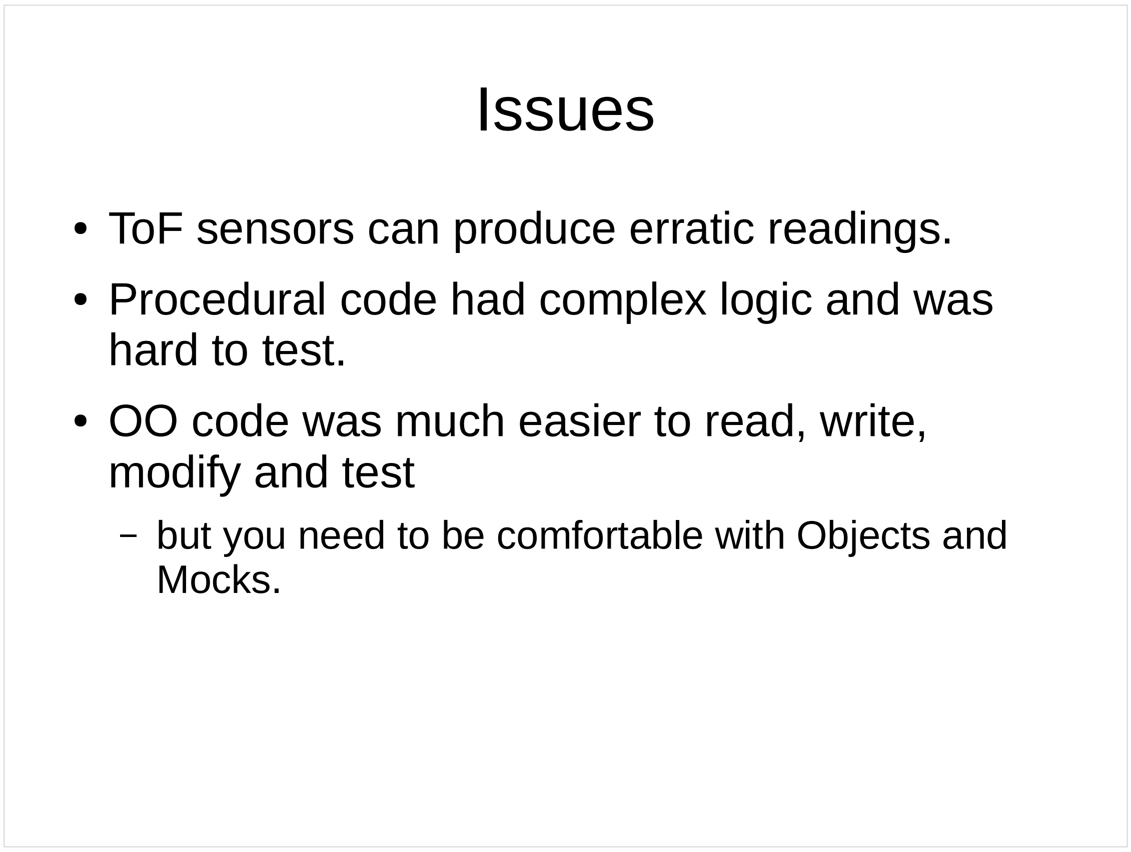
It uses a Raspberry Pi, a Pimoroni Explorer HAT Pro, a buzzer and a VL53L1X ToF sensor.

The ToF sensor produces some spurious readings, so I made a vcouple of chenges in the production version.


B L I N K T !
#18
#22
#24
MOSI
CLK
#13
#19
SDA
SHDN
GND
VIN
SCL
GPIO
2v8
VL53L0X
+3V
Bonnet
Proto



SDA
SCL
RXD
#17
#27
#23
#25
MISO
CE0
#5
#12
#16
#20
#21
#4
#18
#22
#24
MOSI
CLK
CE1
#6
#13
#19
+3V
Bonnet
Proto
VIN
GND
GPIO
SHDN
SCL
SDA
2v8
VL53L0X
fritzing



Issues

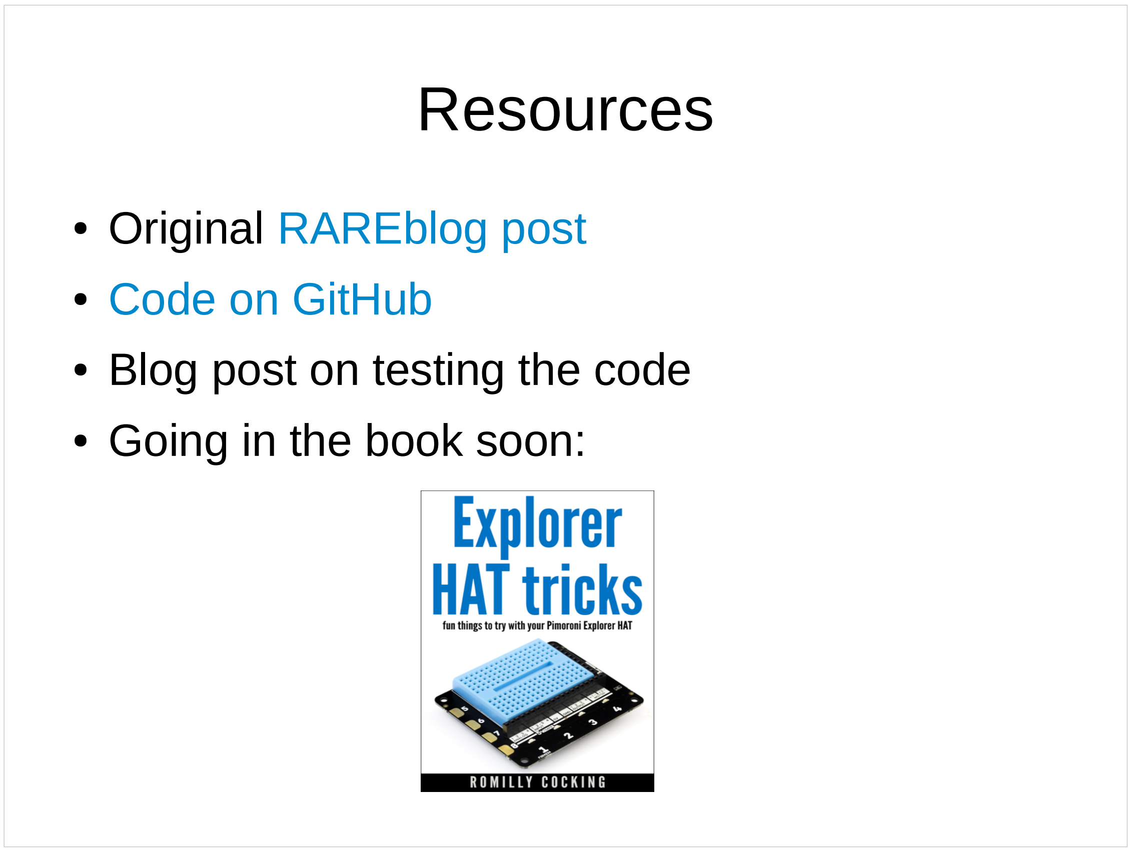
- ToF sensors can produce erratic readings.
- Procedural code had complex logic and was hard to test.
- OO code was much easier to read, write, modify and test
  - but you need to be comfortable with Objects and Mocks.

The production version uses an Adafruit ToF sensor, a Pi Zero WH, and an Adafruit protoBoard.

Credits

Pomodoro: vungoctho (CC BY-SA)

BJ Fogg: Tiny Habits: The Small Changes That Change Everything



Resources

- Original RAREblog post
- Code on GitHub
- Blog post on testing the code
- Going in the book soon:

The code is on GitHub. There are two versions; a procedural implementation, which I *think* works OK, and an O O implementation which I'm almost certain is OK.

The second blog post describes how to test this sort of code using Abstract Classes and Mock Objects.

It makes use of Python type annotations. I use PyCharm for development and the combination helps me spot errors early.