AN INTRODUCTION TO THE I²C SENSOR CONTROLLER (ISC)

Can I use UKMARSBOT sensors with another robot or MCU?

Does your MCU have enough Analogue Input Pins?

Can't I just tell the LEDs to flash and then forget about them?

Is there a simple way to detect Line markers – without false edge triggers?

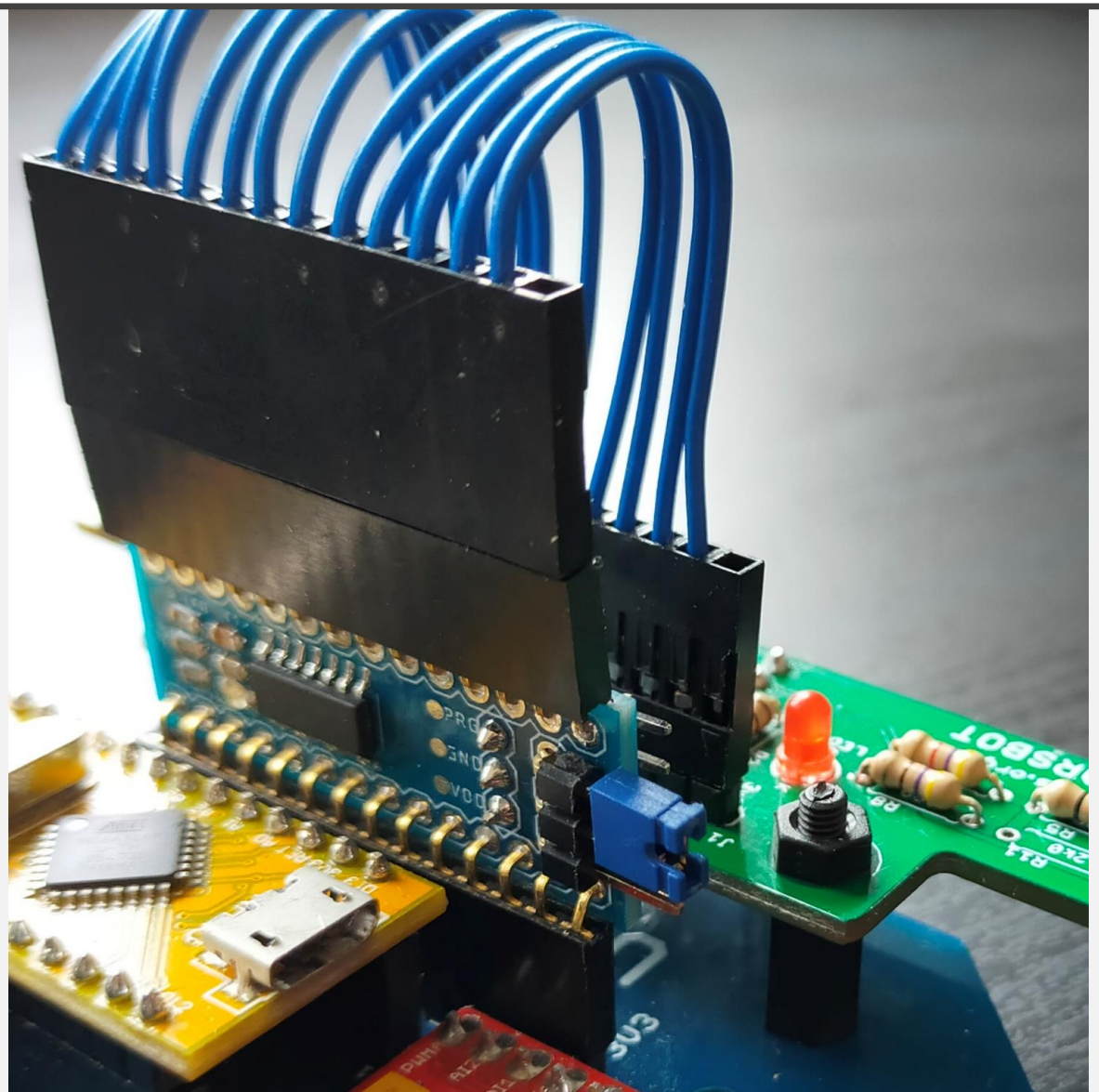
Is there a way to free the I²C bus on UKMARSBOT to use with other peripherals?

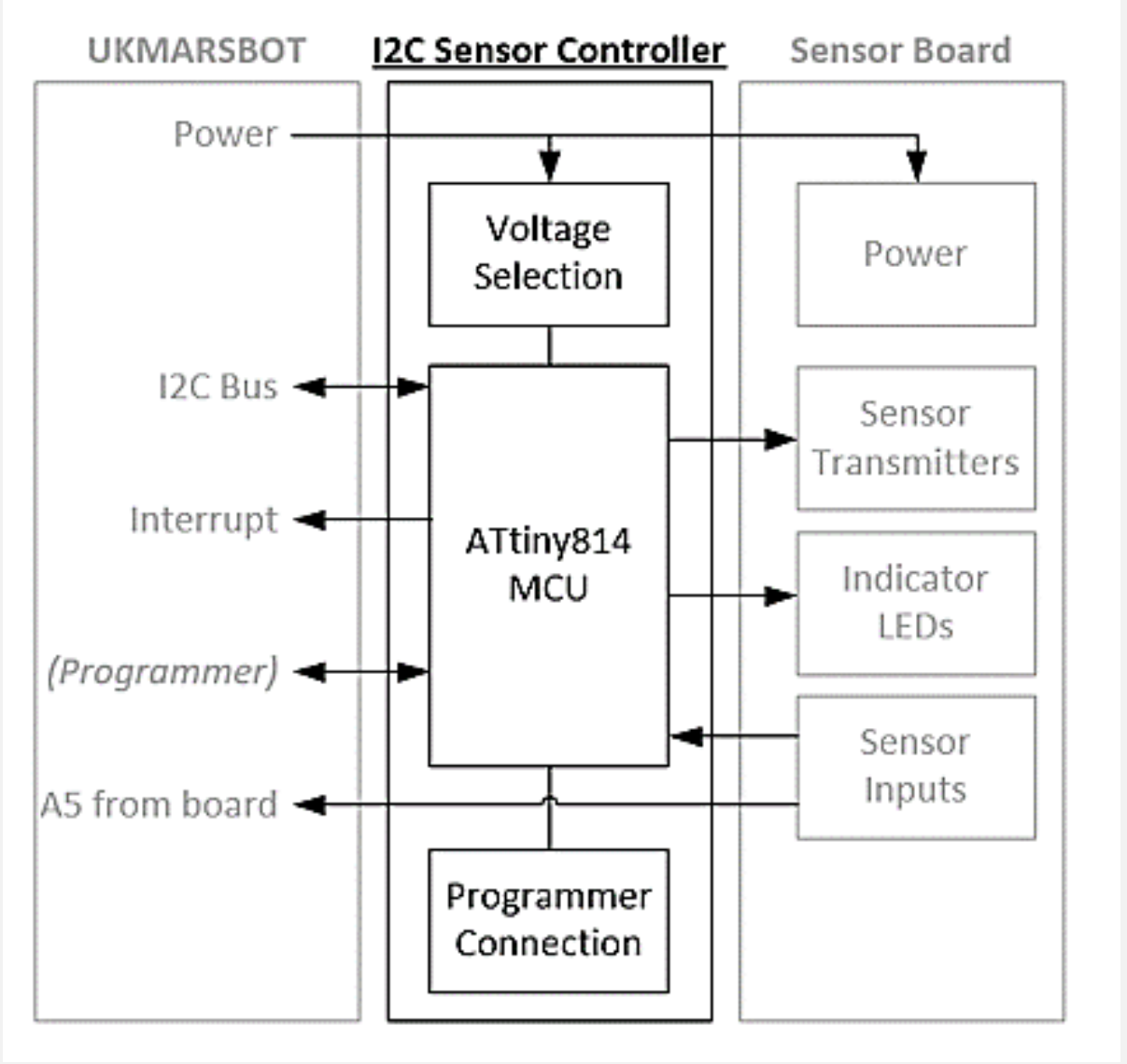
Stephen Pithouse
Presented at Minos 2022 Conference

IN THIS PRESENTATION…

- What is the ISC?
- ISC Specification
- ISC Library
- ISC Examples
- Summary

DROP-IN DEVICE BETWEEN UKMARSBOT AND SENSORS





SPECIFICATION

- Connects directly **between UKMARSBOT and existing UKMARS sensor boards** to offer an I^2C interface for sensors / LED indicators

- Selectable Operating/ADC reference Voltage
  - **3.3V or 5V**

- Supports UKMARS sensor boards that contain up to
  - **5 x Analogue Sensor Inputs** (6 if utilising A5 Analogue Sensor pass-through)
  - **1 x Common Transmitter Output**
  - **2 x Indicator LEDs**

- **Configurable Indicator LED control**
  - Master Control (over I^2C)
  - Sample Frequency output
  - Slow/Fast Blink
  - State Indication
  - Oneshot

SPECIFICATION

- **Collects sensor data periodically**, with configurable:
  - Ambient light removal
  - Sensor transmitter pulse length
  - Resolution (8-bit / 10-bit)
  - Sensor selection – disable unused sensors for increased speed
- **Calibration mode**
  - Returns normalised sensor values
- **Threshold Comparison**
  - Compares each sensor value against a configurable threshold value with hysteresis to return Sensor state
- **Configurable Interrupt Source** (may be polled)
  - Rising and/or Falling Sensor state change
    - Individually Settable for each Sensor

LIBRARY

List of ISC Library Functions

Control / General

init(addr)	-	Reset ISC at given address
enable()	-	Start ISC main routine
isEnabled()	-	Read enabled state of ISC, returns 1 or 0
reset()	-	Drops ISC out of main loop, and resets parameters (powerup state)
read(regAddr, numVals)	-	Read value at regAddr, for numVal registers (1 or 2)
write(regAddr, data, numVals)	-	Write data to regAddr, for numVal registers (1 or 2)
getFirmVer()	-	Read firmware version of ISC
getScanTime()	-	Read time taken for ISC to complete a full cycle

LEDs

setLeftLED(val)	-	Set Left LED mode
setRightLED(val)	-	Set Right LED mode
setBothLEDs(val)	-	Set both LED modes at the same time

Sensors

getSensors()	-	Read all sensors at once, returns outputList[]
getVal(sensor)	-	Read specific sensor, returns value
setPulseDur(val)	-	Set amount of Transmitter on time (in us)
getPulseDur()	-	Read amount of Transmitter on time (in us), returns value
getSensThresh(sens)	-	Set Threshold, above which state is HIGH
setSensThresh(sens, thrsh)	-	Read Threshold, above which state is HIGH, returns value
sensorSetup(sens, val)	-	Set Specific sensor configuration
getSensorSetup(sens)	-	Read specific sensor configuration, returns value
getState(sens)	-	Read all sensor states, returns value
setHysteresis(val)	-	Set Hysteresis (global for all sensors)
getHysteresis()	-	Read Hysteresis value, returns value

Interrupts

enableInterrupt(sens)	-	Enable interrupt for specific sensor
disableInterrupt(sens)	-	Disable interrupt for specific sensor
getInterrupt(sens)	-	Get Interrupt flag state for specific sensor, returns 1 or 0
clearInterrupt(sens)	-	Clear interrupt flag for specific sensor

Micropython Library

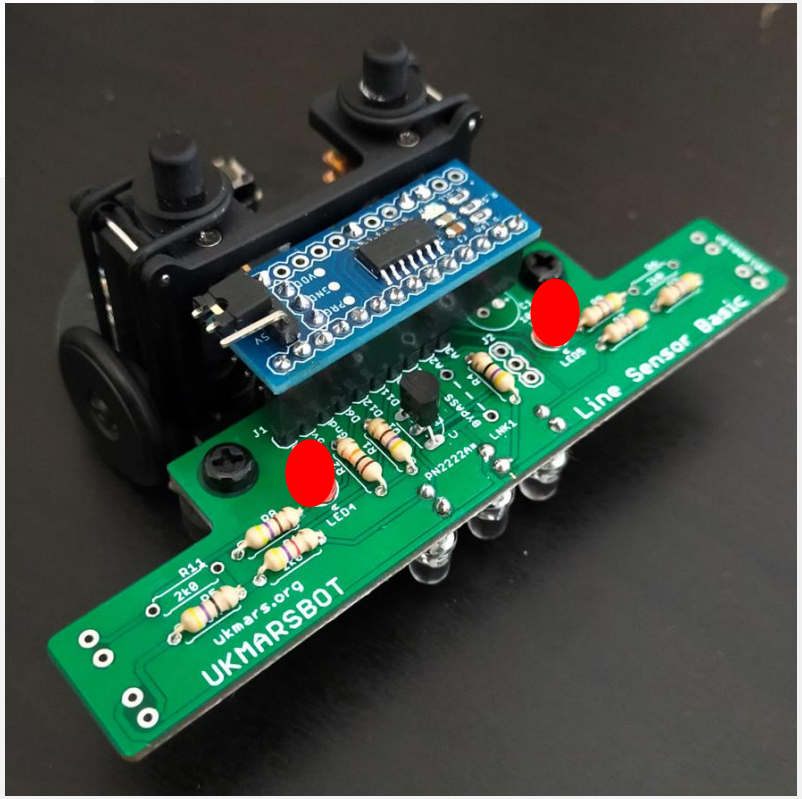
- For this presentation, the Micropython library shall be used

Arduino Library

- Almost exactly the same as Micropython, just C++ syntax

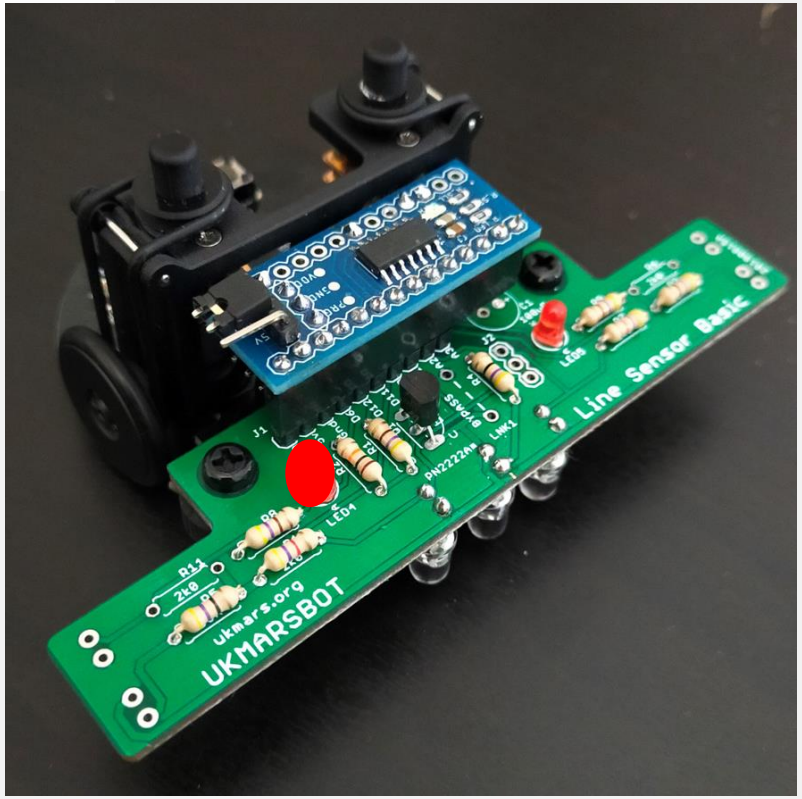
LED INDICATORS

```
1  import ISC                       #Import ISC module to utilise ISC functions
2
3  ISC.init()                       #Connect to ISC and reset it
4  ###Start of ISC Setup ###
5
6  ###End of ISC Setup ###
7  ISC.enable()                     #Start the ISC, once setup is complete
8
9  #### Main code goes here ####
10 ISC.setBothLEDs(ISC.SLOW_BLINK)
```



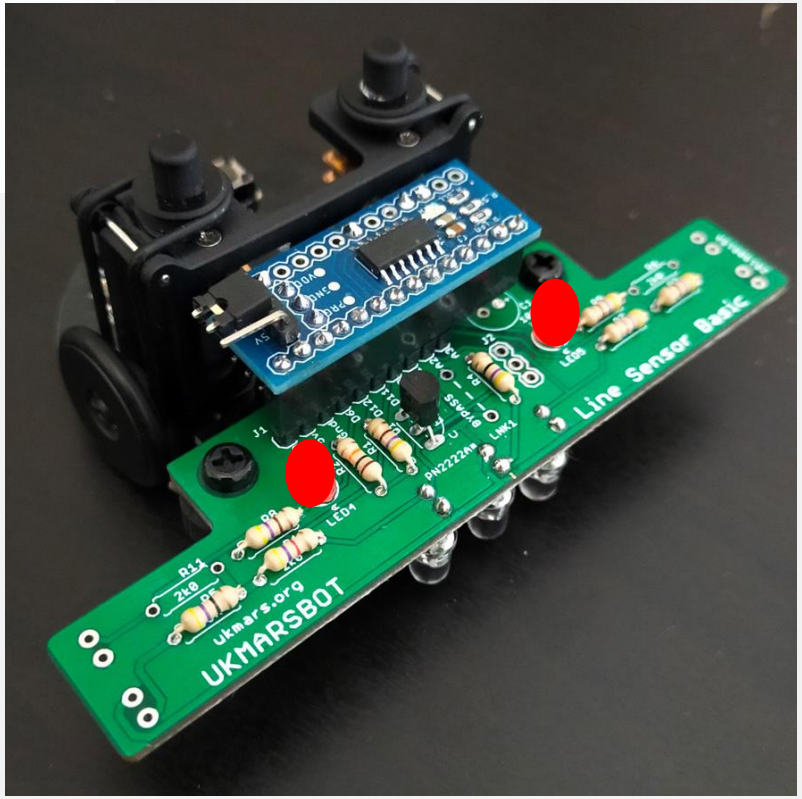
LED INDICATORS

```
1  import ISC                       #Import ISC module to utilise ISC functions
2
3  ISC.init()                       #Connect to ISC and reset it
4  ###Start of ISC Setup ###
5
6  ###End of ISC Setup ###
7  ISC.enable()                     #Start the ISC, once setup is complete
8
9  #### Main code goes here ####
10 ISC.setRightLED(ISC.FAST_BLINK)
```



LED INDICATORS

```
1  import ISC                       #Import ISC module to utilise ISC functions
2
3  ISC.init()                       #Connect to ISC and reset it
4  ###Start of ISC Setup ###
5
6  ###End of ISC Setup ###
7  ISC.enable()                     #Start the ISC, once setup is complete
8
9  #### Main code goes here ####
10 ISC.setLEDs(ISC.SLOW_BLINK, ISC.FAST_BLINK)
```



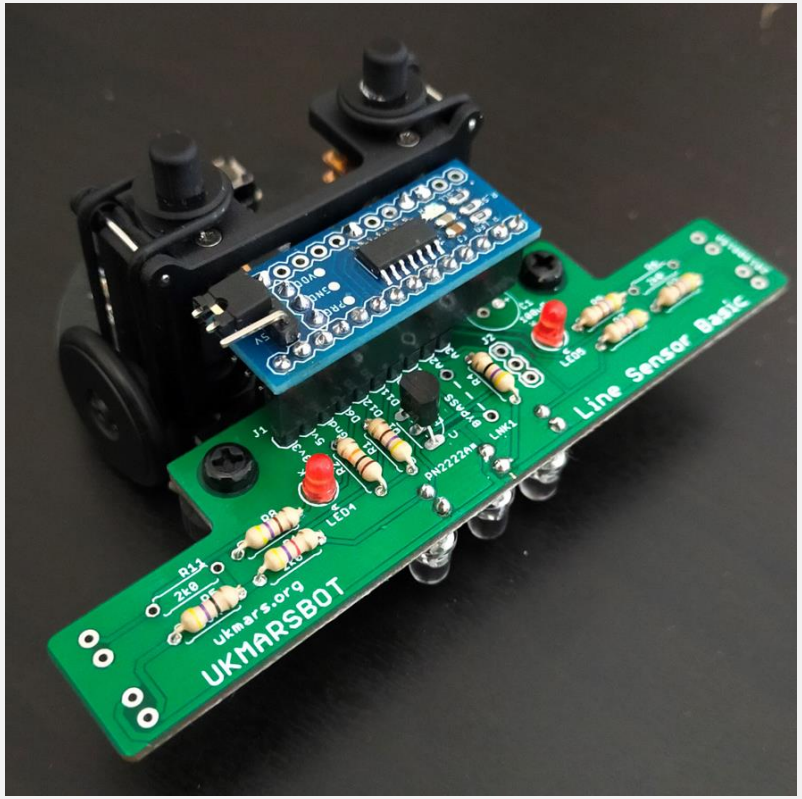
LED INDICATORS

Functions

ISC.setLeftLED()	Control Left LED only (Right remains untouched)
ISC.setRightLED()	Control Right LED only (Left remains untouched)
ISC.setBothLEDs()	Controls Both LEDs

Commands

ISC.OFF	OR	0	Turn LED(s) OFF
ISC.ON	OR	1	Turn LED(s) ON
ISC.TOGGLE			Toggle LED(s) State when the command is executed
ISC.FAST_BLINK			Automatically Toggle LED(s) Fast
ISC.SLOW_BLINK			Automatically Toggle LED(s) State Slow
ISC.ALT_SLOW_BLINK			Inverted version of above
ISC.SENS_0_STATE			Automatically show Sensor state (ON when above threshold)
ISC.SENS_1_STATE			Automatically show Sensor state (ON when above threshold)
ISC.SENS_2_STATE			Automatically show Sensor state (ON when above threshold)
ISC.SENS_3_STATE			Automatically show Sensor state (ON when above threshold)
ISC.SENS_4_STATE			Automatically show Sensor state (ON when above threshold)
ISC.FREQOUT			Toggle LED(s) State each ISC loop – used for diagnostics



READING RAW SENSOR VALUE

```
1  import ISC                          #Import ISC module to utilise ISC functions
2  from time import sleep_ms as delay
3
4  SENSOR_RIGHT_MARK = 0
5
6  ISC.init()                          #Connect to ISC and reset it
7  ###Start of ISC Setup ###
8  ISC.sensorSetup(SENSOR_RIGHT_MARK, ISC.Enable)   #Enable the sensor
9  ###End of ISC Setup ###
10 ISC.enable()                        #Start the ISC, once setup is complete
11
12 ### Main code goes here ####
13 while (True):
14     value = ISC.getVal(SENSOR_RIGHT_MARK)
15     print(value)
16     delay(100)
```

750µs

READING SENSOR VALUE – WITH TX

```
1  import ISC                          #Import ISC module to utilise ISC functions
2  from time import sleep_ms as delay
3
4  SENSOR_RIGHT_MARK = 0
5
6  ISC.init()                          #Connect to ISC and reset it
7  ###Start of ISC Setup ###
8  ISC.sensorSetup(SENSOR_RIGHT_MARK, ISC.Enable + ISC.TxEnb)   #Enable the sensor
9  ###End of ISC Setup ###
10 ISC.enable()                        #Start the ISC, once setup is complete
11
12 ### Main code goes here ####
13 while (True):
14     value = ISC.getVal(SENSOR_RIGHT_MARK)
15     print(value)
16     delay(100)
```

750us

Optionally, change pulse duration by inserting:

```
ISC.setPulseDur(60)  #Tx Pulse Duration (Microseconds)
```

READING NORMALISED SENSOR VALUE

```
1  import ISC                          #Import ISC module to utilise ISC functions
2  from time import sleep_ms as delay
3
4  SENSOR_RIGHT_MARK = 0
5
6  ISC.init()                          #Connect to ISC and reset it
7  ###Start of ISC Setup ###
8  ISC.sensorSetup(SENSOR_RIGHT_MARK, ISC.Enable + ISC.TxEnb)   #Enable the sensor
9  ###End of ISC Setup ###
10 ISC.enable()                        #Start the ISC, once setup is complete
11
12 ISC.startCalibration()                      #Turn Calibration ON
13 delay(4000)                                 #Allow Time to move sensors over the line
14 ISC.endCalibration()                        #Turn Calibration OFF, use normalised values
15
16 ### Main code goes here ####
17 while (True):
18     value = ISC.getVal(SENSOR_RIGHT_MARK)
19     print(value)
20     delay(100)
```

750us

During Calibration: ISC stores Min and Max values for each enabled sensor

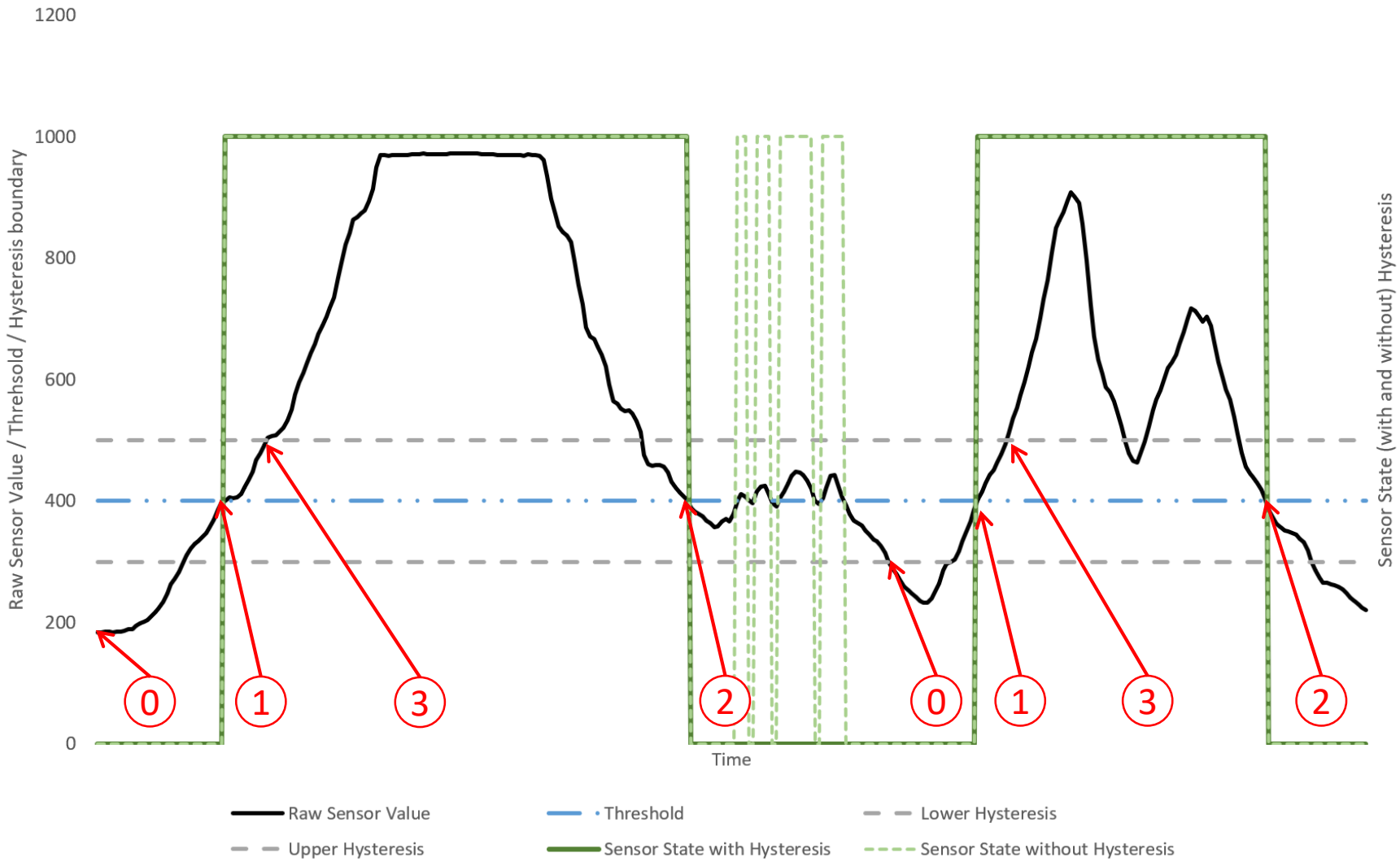
After Calibration: ISC returns normalised values for each sensor

READING SENSOR STATE

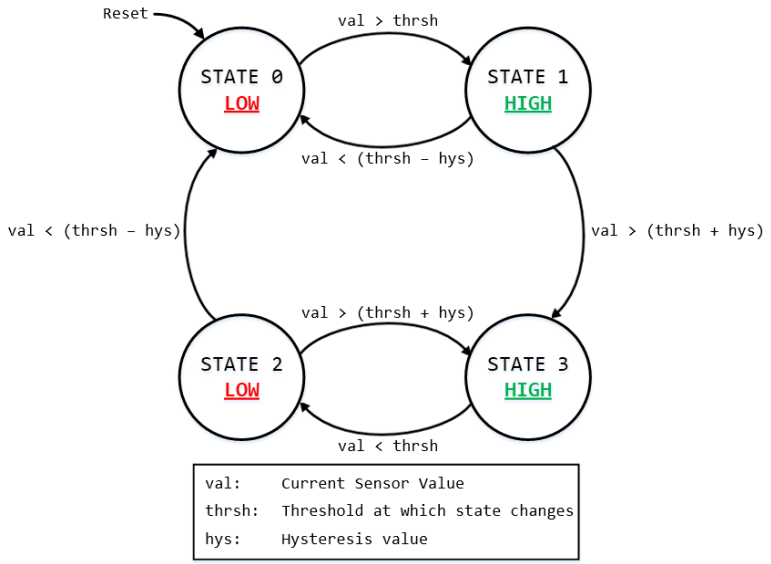
```
1  import ISC                          #Import ISC module to utilise ISC functions
2  from time import sleep_ms as delay
3
4  SENSOR_RIGHT_MARK = 0
5
6  ISC.init()                          #Connect to ISC and reset it
7  ###Start of ISC Setup ###
8  ISC.sensorSetup(SENSOR_RIGHT_MARK, ISC.Enable + ISC.TxEnb)   #Enable the sensor
9  ISC.setSensThresh(SENSOR_RIGHT_MARK, 400)       #Above this value, state = ON
10 ###End of ISC Setup ###
11 ISC.enable()                        #Start the ISC, once setup is complete
12
13 ISC.setRightLED(ISC.stateOf(SENSOR_RIGHT_MARK)) #Indicate state on LED automatically
14
15 ### Main code goes here ####
16 while (True):
17     sensorState = ISC.getState(SENSOR_RIGHT_MARK)
18     print(sensorState)
19     delay(100)
```

720us

SENSOR STATE (WITH HYSTERESIS)



Hysteresis State Diagram



State Number Transition

Note that the state numbering is such that the bitwise AND of state number and 1 gives the state

READING SENSOR STATE – WITH HYSTERESIS

```
1  import ISC                       #Import ISC module to utilise ISC functions
2  from time import sleep_ms as delay
3
4  SENSOR_RIGHT_MARK = 0
5
6  ISC.init()                       #Connect to ISC and reset it
7  ###Start of ISC Setup ###
8  ISC.sensorSetup(SENSOR_RIGHT_MARK, ISC.Enable + ISC.TxEnb)   #Enable the sensor
9  ISC.setSensThresh(SENSOR_RIGHT_MARK, 400)       #Above this value, state = ON
10 ISC.setHysteresis(100)
11 ###End of ISC Setup ###
12 ISC.enable()                     #Start the ISC, once setup is complete
13
14 ISC.setRightLED(ISC.stateOf(SENSOR_RIGHT_MARK)) #Indicate state on LED automatically
15
16 ### Main code goes here ####
17 while (True):
18     sensorState = ISC.getState(SENSOR_RIGHT_MARK)
19     print(sensorState)
20     delay(100)
```

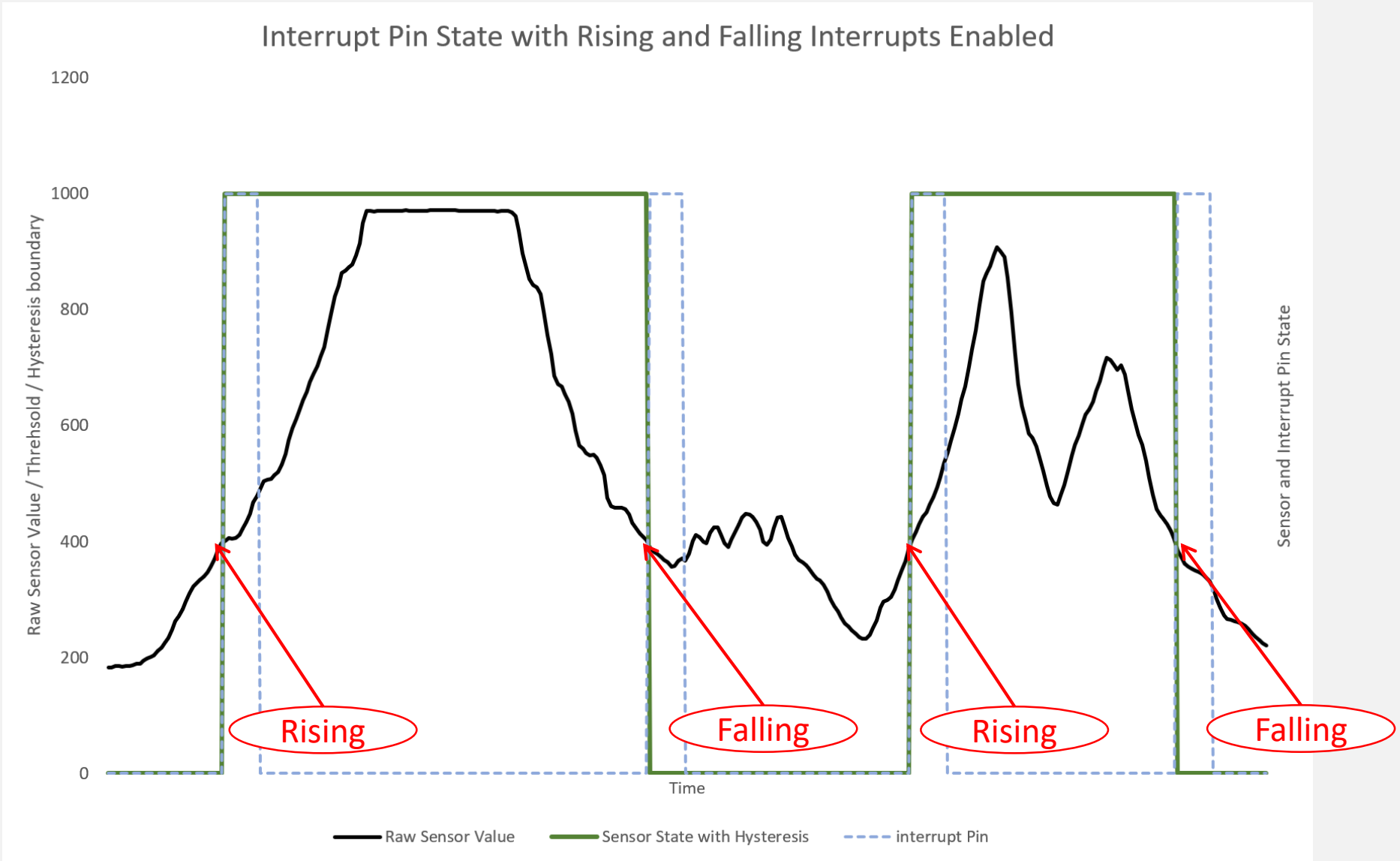
720us

EDGE DETECTION

```
1  import ISC                          #Import ISC module to utilise ISC functions
2  from time import sleep_ms as delay
3
4  SENSOR_RIGHT_MARK = 0
5
6  ISC.init()                          #Connect to ISC and reset it
7  ###Start of ISC Setup ###
8  ISC.sensorSetup(SENSOR_RIGHT_MARK, ISC.Enable + ISC.TxEnb + ISC.Rising)   #Enable the sensor
9  ISC.setSensThresh(SENSOR_RIGHT_MARK, 400)       #Above this value, state = ON
10 ISC.setHysteresis(100)
11 ISC.enableInterrupt(SENSOR_RIGHT_MARK)
12 ###End of ISC Setup ###
13 ISC.enable()                        #Start the ISC, once setup is complete
14
15 ### Main code goes here ####
16 while (True):
17     if(ISC.getInterrupt(SENSOR_RIGHT_MARK)):
18         print("Interrupt")
19         ISC.clearInterrupt(SENSOR_RIGHT_MARK)
20     delay(100)
```

1290us

SENSOR INTERRUPTS



MARKER DETECTION

```
import ISC                      #Import ISC module to utilise ISC functions
from time import sleep_ms as delay

SENSOR_RIGHT_MARK = 0

ISC.init()                      #Connect to ISC and reset it
###Start of ISC Setup ###
ISC.sensorSetup(SENSOR_RIGHT_MARK, ISC.Enable + ISC.TxEnb + ISC.Rising)   #Enable the sensor
ISC.setSensThresh(SENSOR_RIGHT_MARK, 400)       #Above this value, state = ON
ISC.setHysteresis(100)
ISC.enableInterrupt(SENSOR_RIGHT_MARK)
###End of ISC Setup ###
ISC.enable()                    #Start the ISC, once setup is complete

ISC.clearInterrupt(SENSOR_RIGHT_MARK)     #Clear any initial interrupt flag
lineCount = 0

### Main code goes here ####
while (True):
    if(ISC.intPin.value()):      #Only look at interrupt checks if interrupt input is on
        if(ISC.getInterrupt(SENSOR_RIGHT_MARK)):
            lineCount+=1
            print(lineCount)
            ISC.clearInterrupt(SENSOR_RIGHT_MARK)
            ISC.setBothLEDs(ISC.TOGGLE)
    delay(100)
```

READING MULTIPLE SENSORS

```
1  import ISC                      #Import ISC module to utilise ISC functions
2  from time import sleep_ms as delay
3
4  ISC.init()                      #Connect to ISC and reset it
5  ###Start of ISC Setup ###
6  ISC.init()                                              #Connect to ISC, reset it, initialising all values
7  ISC.sensorSetup(0, ISC.Enable + ISC.TxEnb + ISC.Rising) #Enable the sensor, with rising edge detection for Start / Stop Marker Interrupt
8  ISC.sensorSetup(1, ISC.Enable + ISC.TxEnb)              #Enable the sensor
9  ISC.sensorSetup(2, ISC.Enable + ISC.TxEnb)              #Enable the sensor
10 ISC.sensorSetup(3, ISC.Enable + ISC.TxEnb)              #Enable the sensor
11 ISC.sensorSetup(4, ISC.Enable + ISC.TxEnb)              #Enable the sensor
12 ISC.enableInterrupt(0)                                  #Enable Interrupt for Start / Stop Marker Sensor
13 ###End of ISC Setup ###
14 ISC.enable()                    #Start the ISC, once setup is complete
15
16 sensVal = [0, 0, 0, 0, 0]
17
18 ### Main code goes here ####
19 while (True):
20     sensVal = ISC.getSensors()     #Read all sensor values from ISC
21     print(sensVal)
22     delay(100)
```

Output:

```
[77, 28, 67, 65, 19]
[76, 27, 66, 68, 20]
[76, 26, 65, 65, 20]
[75, 27, 64, 64, 19]
```

1570us

OPTIMISED FUNCTIONALITY

```
import ISC                      #Import ISC module to utilise ISC functions
from time import sleep_ms as delay

ISC.init()                      #Connect to ISC and reset it
###Start of ISC Setup ###
ISC.init()                                              #Connect to ISC, reset it, initialising all values
ISC.sensorSetup(0, ISC.Enable + ISC.TxEnb + ISC.Rising) #Enable the sensor, with rising edge detection for Start / Stop Marker Interrupt
ISC.sensorSetup(1, ISC.Enable + ISC.TxEnb)              #Enable the sensor
ISC.sensorSetup(2, ISC.Enable + ISC.TxEnb)              #Enable the sensor
ISC.sensorSetup(3, ISC.Enable + ISC.TxEnb)              #Enable the sensor
ISC.sensorSetup(4, ISC.Enable + ISC.TxEnb)              #Enable the sensor
ISC.enableInterrupt(0)                                  #Enable Interrupt for Start / Stop Marker Sensor
###End of ISC Setup ###
ISC.enable()                    #Start the ISC, once setup is complete

sensVal = [0, 0, 0, 0, 0]

### Main code goes here ####
while (True):
    sensVal = ISC.getSensorsLowRes()     #Read all sensor values from ISC
    print(sensVal)
    delay(100)
```

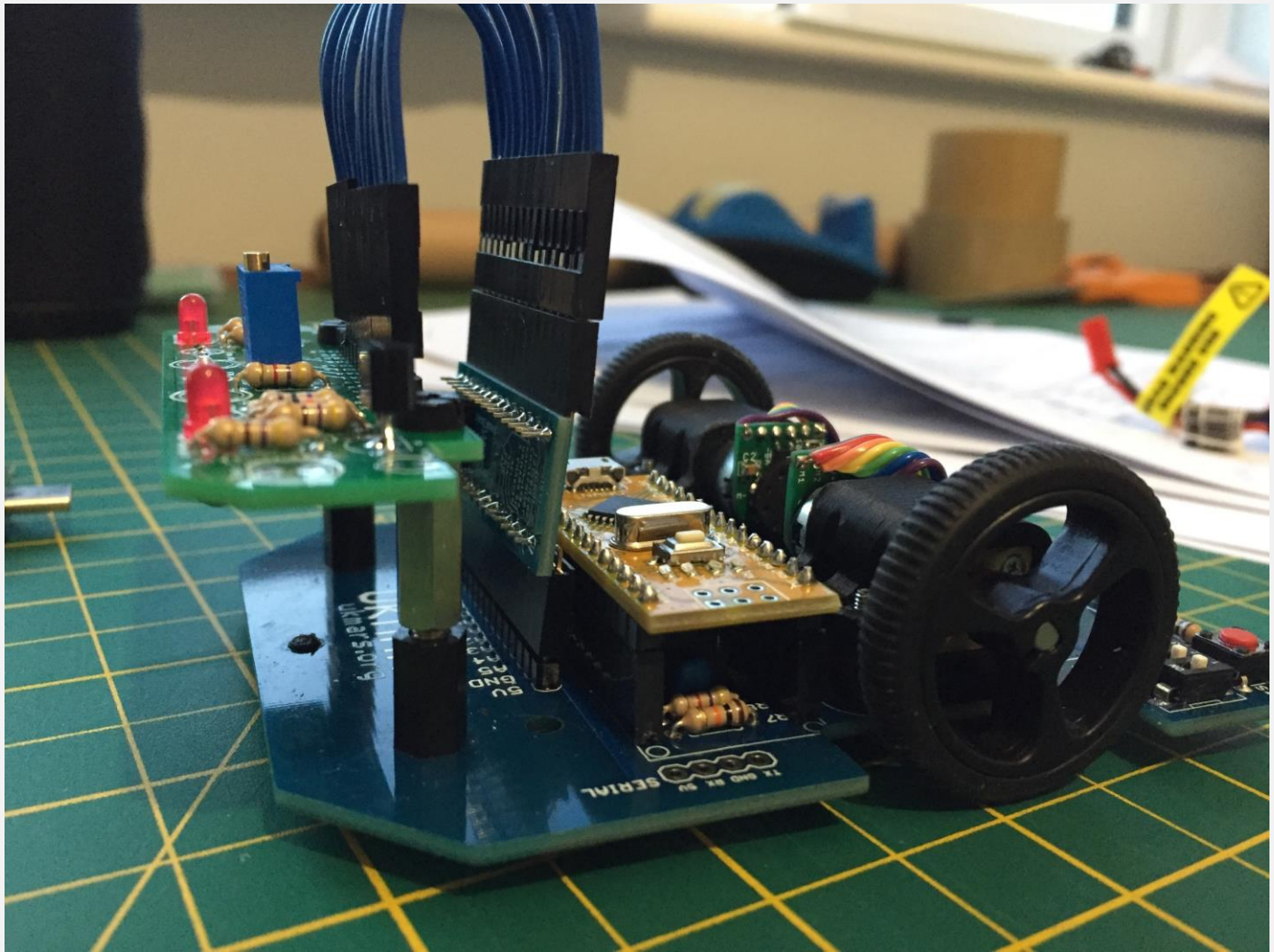
1020us

REGISTER SUMMARY

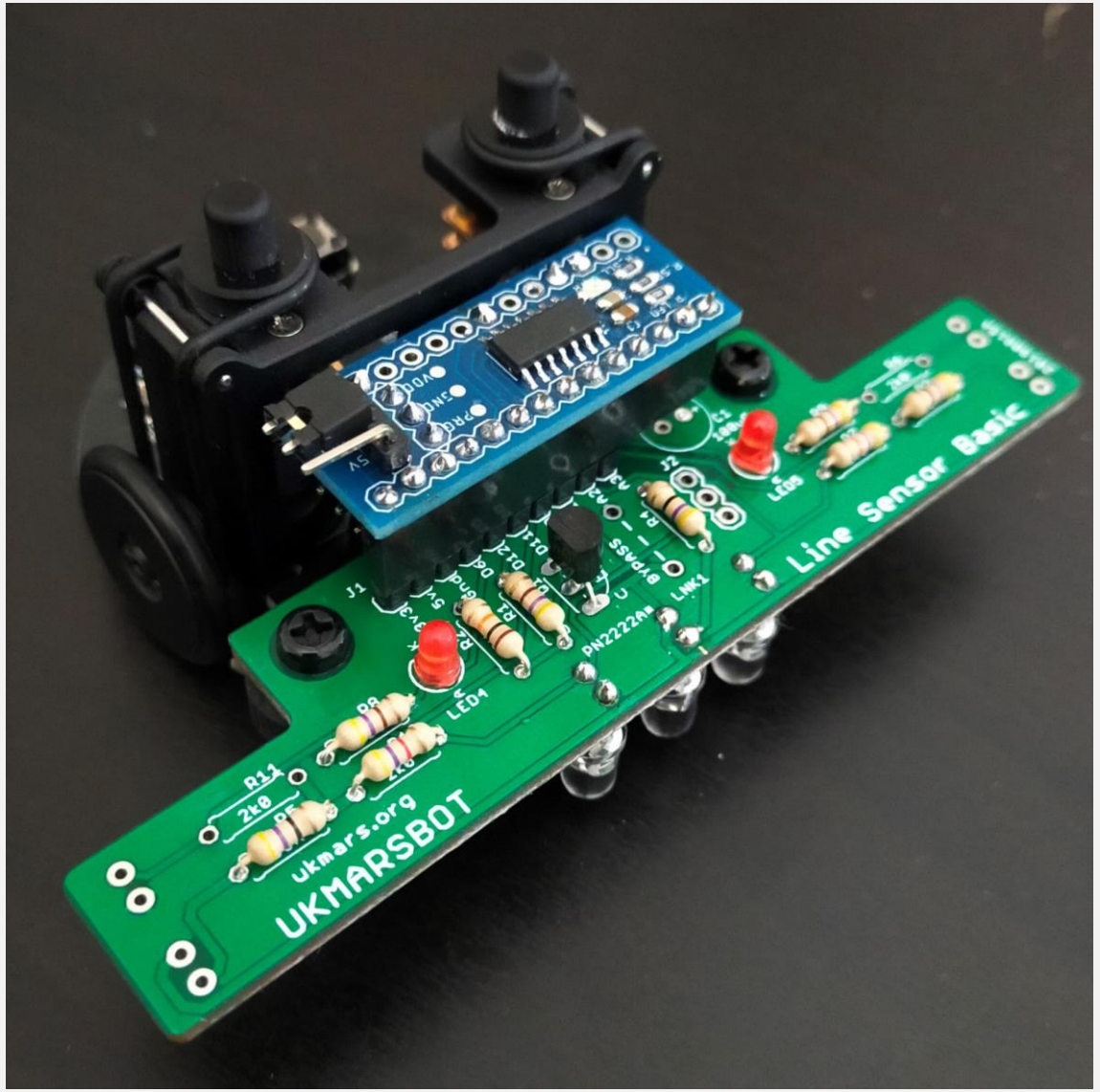
6.0 ISC REGISTER SUMMARY

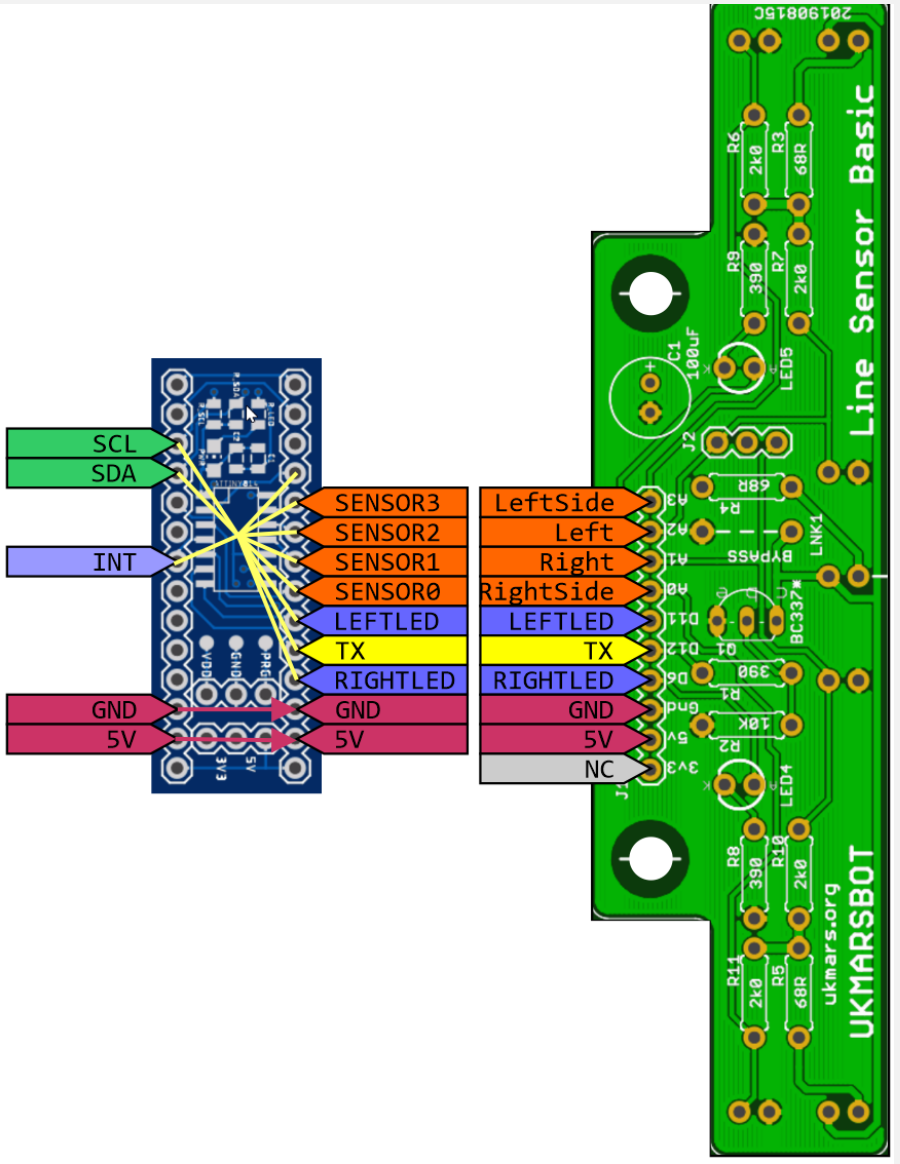
OFFSET	Name	Bit Pos									
0 (0x00)	FIRMVER	7:0	FIRMWARE_VERSION								
1 (0x01)	CONTROL	7:0	ENABLED			BOARD TYPE				RESET	ENABLE
2 (0x02)	INTERRUPT CTRL	7:0	BRDINTEN			SENS4 INTEN	SENS3 INTEN	SENS2 INTEN		SENS1 INTEN	SENS0 INTEN
3 (0x03)	INTERRUPT FLAGS	7:0	BRDINT			SENS4 INT	SENS3 INT	SENS2 INT		SENS1 INT	SENS0 INT
4 (0x04)	SENSSTATE	7:0				SENS4 STATE	SENS3 STATE	SENS2 STATE		SENS1 STATE	SENS0 STATE
5 (0x05)	LEDCTRL	7:0	LLED	RLED				INDICATION_MODE			
6 (0x06)	PULSEDUR	7:0	PULSE_DURATION (µs)								
7 (0x07)	SENS0SETUP	7:0	R_INTENB	F_INTENB	POLARITY			TX_EN		RESOL	ENB
8 (0x08)	SENS1SETUP	7:0	R_INTENB	F_INTENB	POLARITY			TX_EN		RESOL	ENB
9 (0x09)	SENS2SETUP	7:0	R_INTENB	F_INTENB	POLARITY			TX_EN		RESOL	ENB
10 (0x0A)	SENS3SETUP	7:0	R_INTENB	F_INTENB	POLARITY			TX_EN		RESOL	ENB
11 (0x0B)	SENS4SETUP	7:0	R_INTENB	F_INTENB	POLARITY			TX_EN		RESOL	ENB
12 (0x0C)	Reserved	7:0									
13 (0x0D)	SENS0VAL	15:8	MSByte of SENS0VAL								
14 (0x0E)		7:0	LSByte of SENS0VAL								
15 (0x0F)	SENS1VAL	15:8	MSByte of SENS1VAL								
16 (0x10)		7:0	LSByte of SENS1VAL								
17 (0x11)	SENS2VAL	15:8	MSByte of SENS2VAL								
18 (0x12)		7:0	LSByte of SENS2VAL								
19 (0x13)	SENS3VAL	15:8	MSByte of SENS3VAL								
20 (0x14)		7:0	LSByte of SENS3VAL								
21 (0x15)	SENS4VAL	15:8	MSByte of SENS4VAL								
22 (0x16)		7:0	LSByte of SENS4VAL								
23 (0x17)	Reserved	15:8									
24 (0x18)		7:0									
25 (0x19)	SENS0THRSH	15:8	MSByte of SENS0THRSH								
26 (0x1A)		7:0	LSByte of SENS0THRSH								
27 (0x1B)	SENS1THRSH	15:8	MSByte of SENS1THRSH								
28 (0x1C)		7:0	LSByte of SENS1THRSH								
29 (0x1D)	SENS2THRSH	15:8	MSByte of SENS2THRSH								

APPLICATION: ARDUINO UKMARSBOT

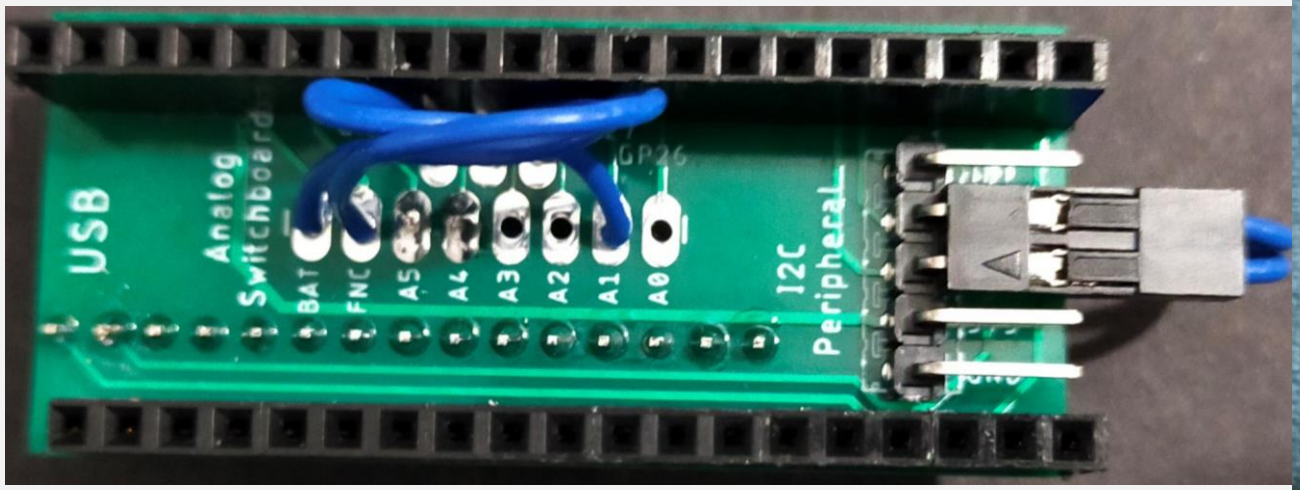


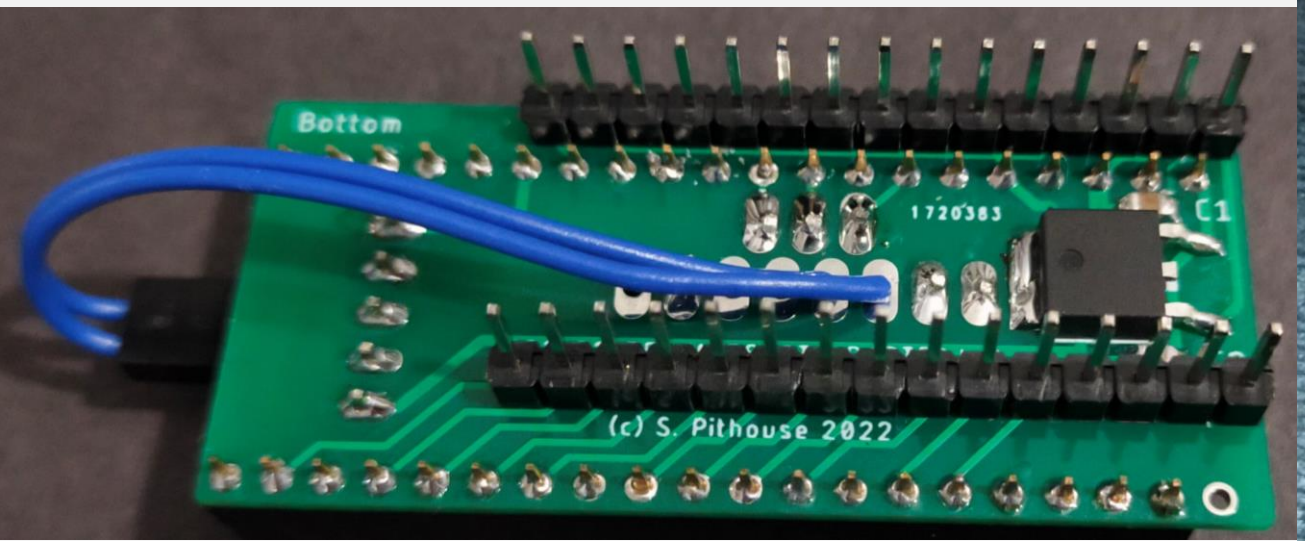
APPLICATION – LINE SENSOR ON CUSTOM MOUSE

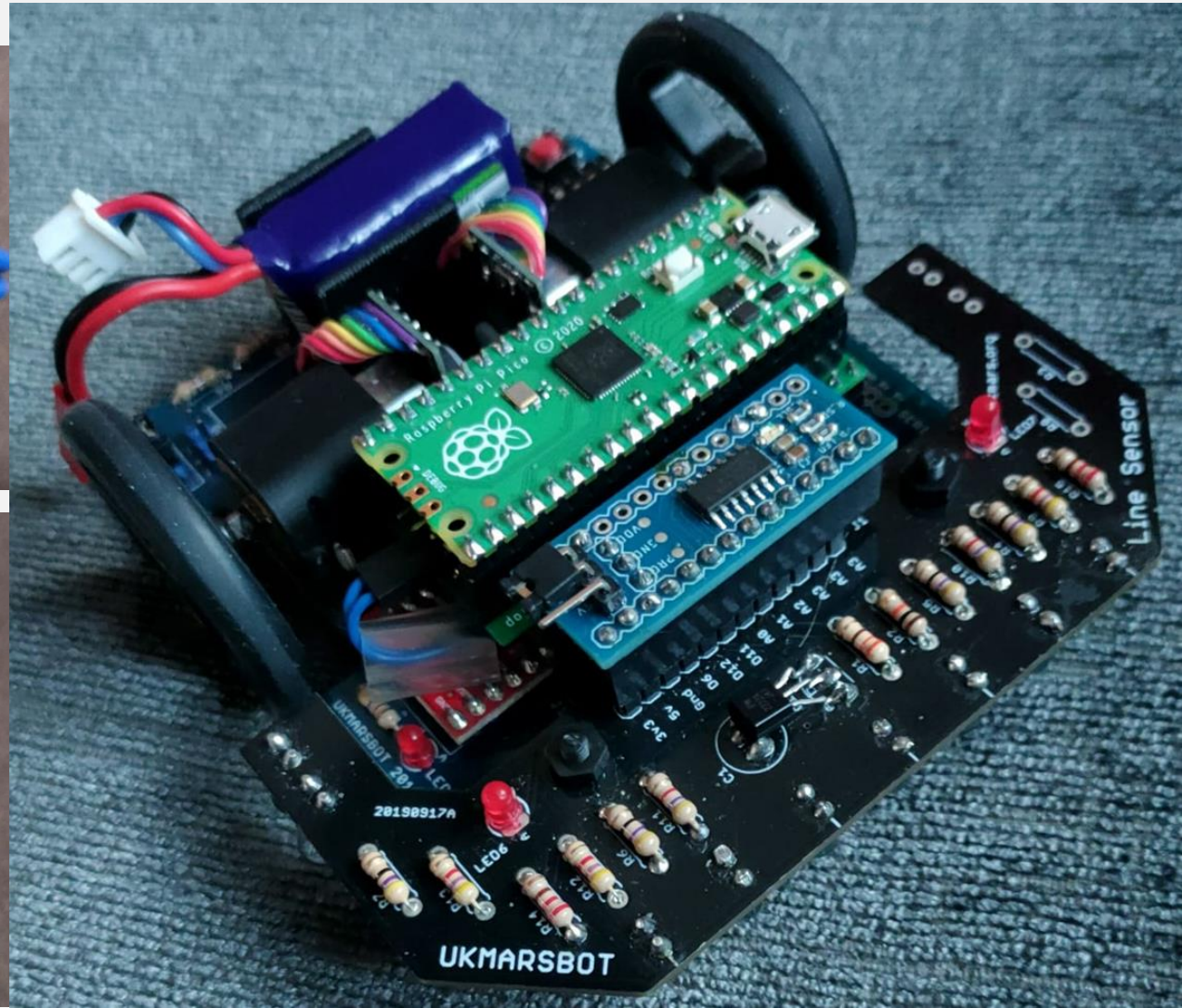




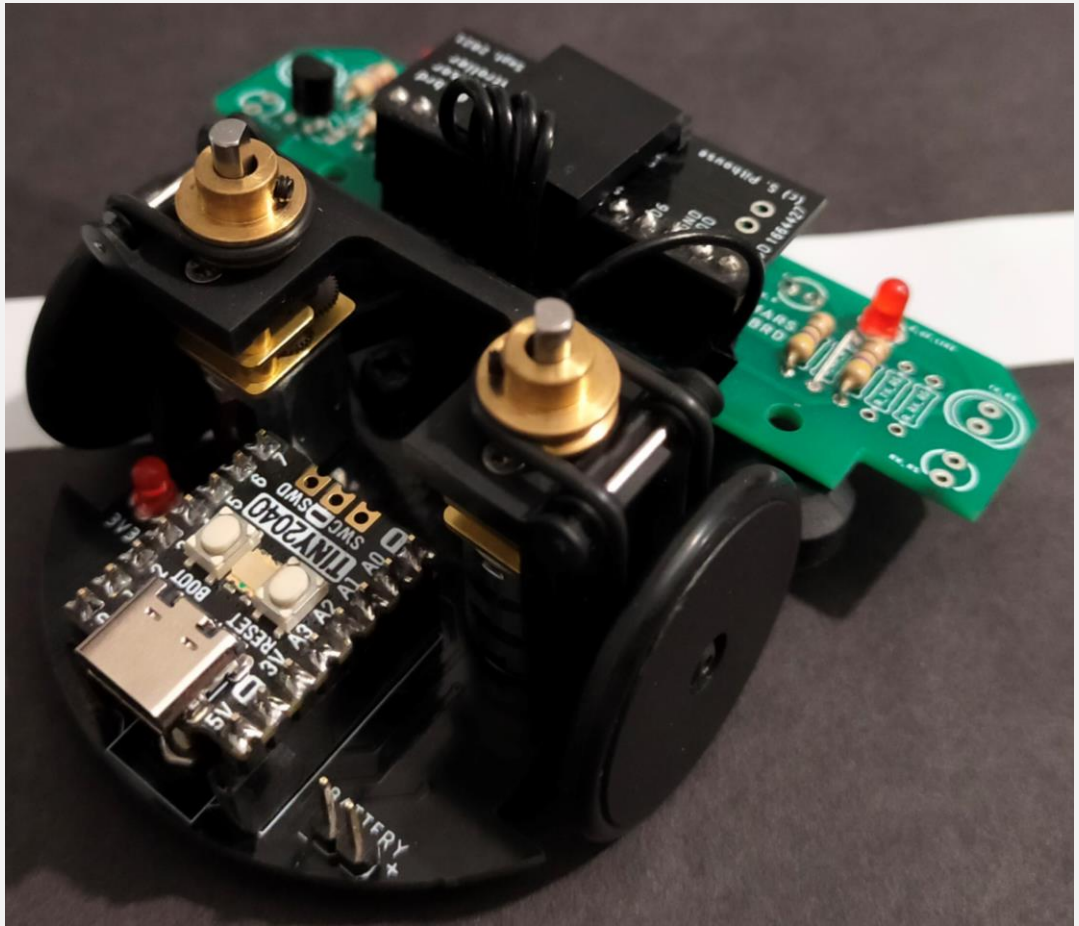
APPLICATION - RPI PICO UKMARSBOT







APPLICATION - RPI TINY2040 HALF-SIZE

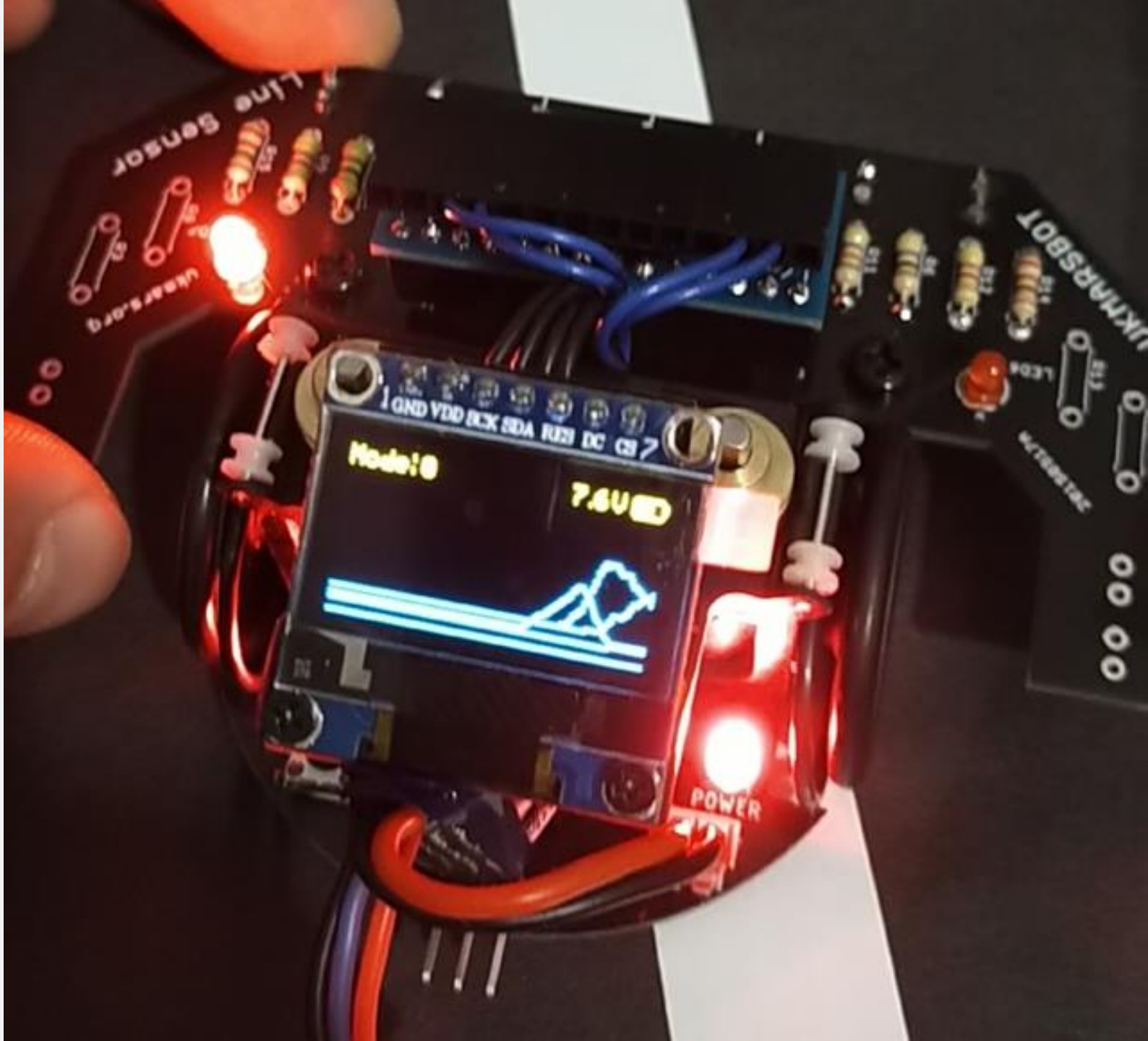


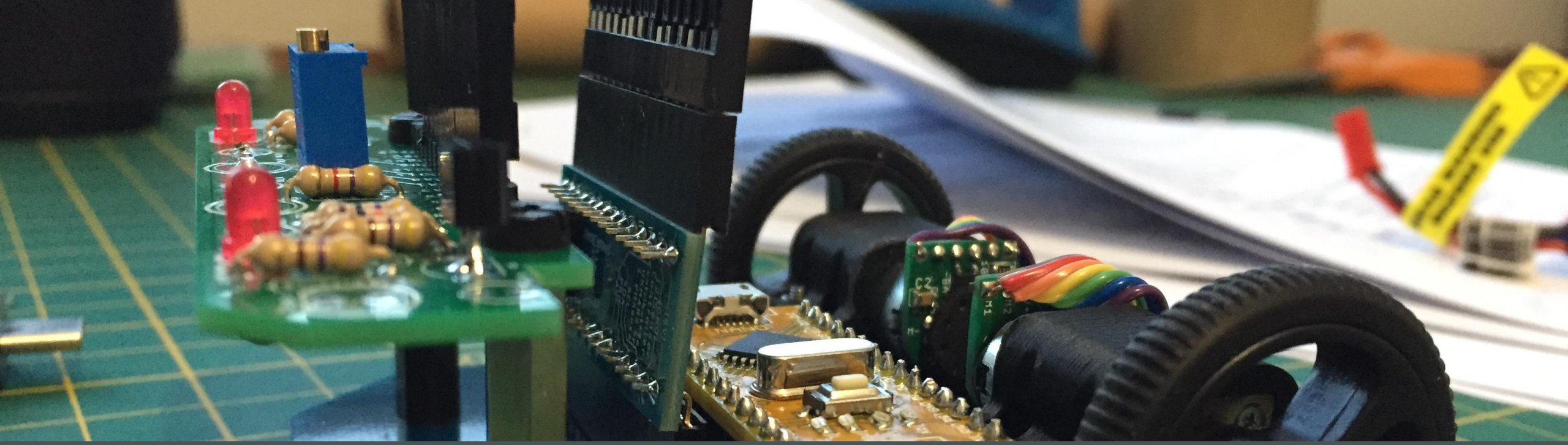
SUMMARY

- Simplifies Arduino Sensor Code
  - Start/Stop marker detection by using configurable interrupts and internal hysteresis to prevent nuisance trips
- Simplifies Arduino LED Indicator Code
- Frees up Analog Inputs for other tasks
- Facilitates the use of the I2C bus on the MARSBOT
  - Allowing Accelerometer / displays to be fitted
- I2C interface makes sensor boards compatible with MCUs with no Analog Inputs.

NEXT STEPS

- Increase sensors to 6
- Could convert to SPI – but not useful for UKMARSBOT
- Global Parts Shortage…
- Any interest?
- Any comments?





SUPPORTING DOCUMENTATION

- ISC Datasheet
- ISC Library Documentation
- Python Library and Example Programs
- Arduino Library and Example Sketches
- Github link: https://github.com/slinkyfish/ISC-Project