Design note: Pico Robot

Adrie Huesman
Start date: 1 Nov 2022
Last update: 12 Dec 2022

1. Introduction

Pico robot is a small (10 by 15 cm) obstacle avoiding robot. There are a few new aspects of this robot:

1. The use of a RP2040 microcontroller (32 bits). This chip is typically used on a Pico development board, hence the robots name. However, in this case a Maker Pi RP2040 robotics board is used.
2. The use of MicroPython and Thonny for programming. The main design question is here: Is MicroPython powerful (IO, Sequential Function Chart) and fast (encoders) enough to support the required functionality?

Figure 1 shows a photo of Pico Robot.

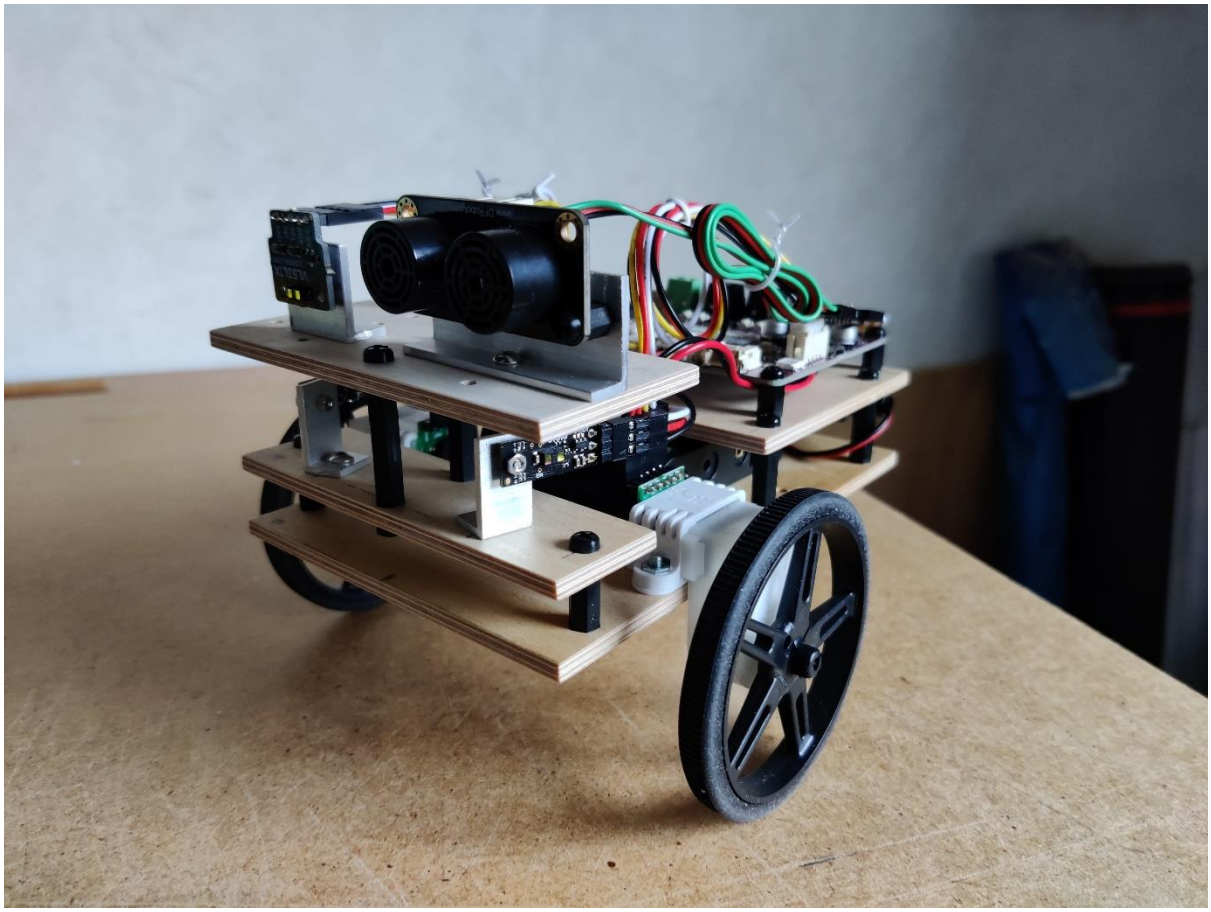


Figure 1 Pico robot. Note the IR and US sensor on top and the small IR distance sensors below.

2. Design details

The figure below gives an overview of the (planned) electric/electronic hardware. Table 1 gives an over of the specific GPIO usage.

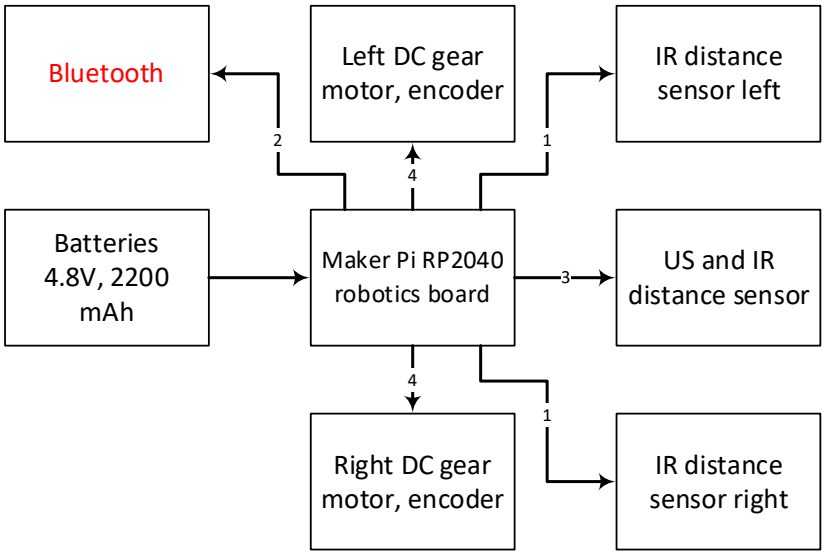


Figure 2 A block diagram of the electronic hardware. The numbers in the arrows indicated how many GPIO are involved (15 in total). Modules in red have not been added yet.

Table 1 Usage of the Maker Pi RP2040 GPIO. Future changes are indicated in red.

GPIO	Type	Use (rev G)	Use (future)
0	Grove port 1	Right encoder	Right encoder
1	Grove port 1	Right encoder	Right encoder
2	Grove port 2		US sensor trig/echo
3	Grove port 2		
4	Grove port 3		Bluetooth TX
5	Grove port 3	US sensor tri/echo	Bluetooth RX
6	Grove port 5	Left IR sensor	Left IR sensor
7	Grove port 7	Left encoder	Left encoder
8	M1A	M1A	M1A
9	M1B	M1B	M1B
10	M2A	M2A	M2A
11	M2B	M2B	M2B
12	Servo port		
13	Servo port		
14	Servo port		
15	Servo port		
16	Grove port 4	VL53L1X I2C sda	VL53L1X I2C sda
17	Grove port 4	VL53L1X I2C scl	VL53L1X I2C scl
18	WS2812B neopixels	WS2812B neopixels	WS2812B neopixels
20	Programmable Button	Programmable Button	Programmable Button
21	Programmable Button	Programmable Button	Programmable Button
22	Buzzer	Buzzer	Buzzer
26	Grove port 5/6	Right IR sensor	Right IR sensor
27	Grove port 6		
28	Grove port 7	Left encoder	Left encoder

2.1 Drive

A 2-wheel differential drive with swivel wheel was selected. The motors are Pololu Mini Plastic Gearmotors. The robotics board provides the motor drivers. Two standard 70 mm Pololu wheels are used. The encoders indicate three steps down per motor revolution. So, the resolution is (70 × pi)/(120 × 3) = 0.61 mm/step (120 is the gear ratio). Given a wheel distance of 137 mm this implies (0.25 × pi × 137)/0.61 = 176 steps for a 90-degree turn.

2.2 Chassis

The chassis was made of 4 mm aviation plywood. It measures 15 by 10 cm and there are three levels. The lower level is reserved for the motors and the batteries, the upper levels are for the controller and sensors. The upper level consists of three parts. So, for drilling extra holes only part of the robot needs to be disassembled.

2.3 Battery and converter

The power comes from 4 AA NiMH batteries. The 4.8 volt provided is within specification for all motors. The power switch, 3 3v step down converter etc. are provided by the robotics board.

2.4 Microcontroller

As controller a Maker Pi RP2040 robotics board was chosen. It is adequate (enough IO and other resources), small, cost-effective and supports an increasing number of libraries! Connections are made via Grove connectors. The board can be programmed via CircuitPython, MicroPython, C/C++ and Arduino. MicroPython was selected since it seems adequate (external interrupts are supported) and to try something new.

2.5 Sensors

The following sensors are used:

- Magnetic Encoder Pair Kit, 12 CPR, 2.7-18V: https://www.pololu.com/product/1523
- Ultrasonic sensor DFRobot Gravity URM09: https://wiki.dfrobot.com/URM09_Ultrasonic_Sensor_Gravity_Trig_SKU_SEN0388
- VL53L1X Time of Flight (ToF) Sensor: https://shop.pimoroni.com/products/vl53l1x-breakout?variant=12628497236051
- 2 Pololu Distance Sensors with Pulse Width Output: https://www.pololu.com/product/4064 or https://www.pololu.com/product/4079

The Field of View (FoV) was increased by placing the US and ToF sensor next to each other (see figure 1). In experiments, see appendix C, the FoV was determined to be around 20 and 30 degrees for the ToF and US sensor respectively. Note that these FoVs translate @ 30 cm into a 8+5+5 = 18 cm wide obstacle detection range, while the robot width including wheels is around 15 cm. Furthermore, the combination of two different measuring principles offers the advantage that the robot can detect dark and soft obstacles.

2.6 Actuators

The robot has three types of actuators:

- Buzzer (provided by the robotics board).
- Neopixels (provided by the robotics board).

- 2 120:1 Mini Plastic Gearmotors HP: https://www.pololu.com/product/1520

2.7 Software

The software was written/developed in MicroPython. From a control point of view the software are implementations of proportional control and a Sequential Function Charts (SFC). To facilitate using the various sensors and actuators several Functions were written like “move”, see appendix A. The SFC that shapes the overall obstacle avoidance is shown in figure 3. The essence is that distance measurements are done or reconfirmed in the SAM state, in other words while to robot doesn’t move. Distance measurements are best done when there are no vibrations.

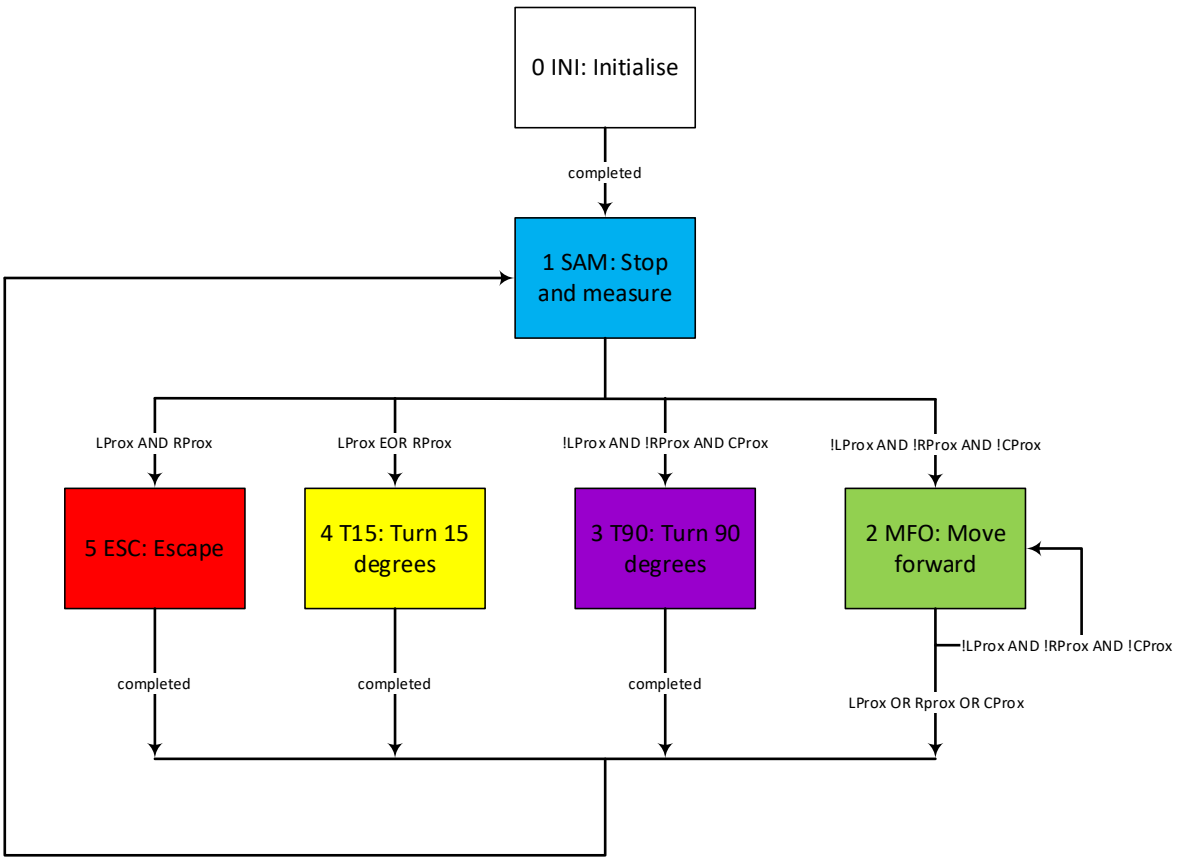


Figure 3: The sequential function chart for obstacle avoidance. By changing color, the neopixels indicate the current state of the robot, for example in the MFO state the neopixels turn green.

Below is some further explanation on the move function and the MFO state:

- Move uses high gain proportional control that keeps the left and right steps very close together (< 3 steps). For that reason, only the leftsteps is checked to stop: `while leftsteps < steps:`
- The MFO state also makes use of high gain proportional control. Note that only every 85 steps (around 5 cm) distance measurements are taken: `if (leftsteps - lastmeas) > 85:` This is to avoid faulty sensor measurements during acceleration and allows for proportional control to be executed at high frequency.

3. Performance

The performance of the robot is very good, certainly comparable with a robot programmed in Arduino. Occasionally on carpet the robot seems to veer of somewhat to the left of right. This may have to do with the fact that one wheel sinks further in the carpet. This tendency seems absent on hard floors.

The total current drawn by the robot depends on its state:

- INI: 50 mA.
- MFO: 250 - 350 mA depending on motor speed.

4. Conclusions and follow-up

- MicroPython is powerful and fast. All needed functionalities could be programmed. The encoders can generate frequencies up to 900 Hz. This can be handled via external interrupts without any problem.
- MicroPython together with Thonny provides a simple, pleasant, and fast IDE. It is easy to work on multiple files and to test pieces of code like functions. In addition, there is no compilation time.
- The Maker Pi RP2040 robotics board is really a nice board and very well suited to build small robots. Some suggestions for improvement:
    - More Grove connectors (say in total eight) included in the box.
    - ~~Orient all Grove connectors vertical (like 1 & 7); this makes it much easier to (dis)connect.~~
    - USB-C connector rather than micro-USB.
    - Via an extra switch make the + pin of the 4 servo ports selectable between 3.3v and VIN.
- There were no GPIO "surprises" like for ESP32 microcontrollers. The only surprise was that after importing the libraries a 2 second delay was needed to boot up on batteries.
- The FoV can be increased by:
    - Sensor selection based on FoV.
    - Sensor location and/or orientation, for example moving sensors to the back.
    - Using more sensors (sensor array), for example a US sensor with an IR sensor. If sensors with different measurement principles are used more robust performance is obtained (black or soft surfaces).
- The FoV and stopping distance for T90 (now 30 cm) are related for good performance:
    - obstacle detection range < width of the robot; possible obstacle collision.
    - obstacle detection range >= width of the robot; correct obstacle avoidance.
    - obstacle detection range >> width of the robot; conservative obstacle avoidance.

- ~~Streamline the code; check need for pull down/up.~~
- ~~Slow down the movement of the servo.~~
- ~~Consider adding ramp-up and ramp-down in MFO state.~~
- ~~Implement a smooth stop in the MFO state.~~
- Consider adding ramp-up and ramp-down in move function.
- Add a Bluetooth module (HC-05, HC-08 etc.).
- ~~Try collection of data in a file.~~
- ~~Increase the FOV e.g., add a dynamic IR proximity sensor or make other design changes.~~
- Examine the explorino robot code: https://github.com/giomalt/explorino.

From a RP2020 point of view the following is interesting:

- Try using the second core, e.g., for taking measurements. This can be done in MicroPython.
- ~~Try analog inputs, e.g., the internal temperature measurement.~~
- ~~Try I2C, e.g., with a Time Of Flight (TOF) sensor VL53L1X (how to deal with multiple sensors?) or a display.~~
- Try the Programmable IO (PIO). This seems very interesting to deal with the encoders. See the get started book and https://blues.io/blog/raspberry-pi-pico-pio/ for initial information.

5. References

1. Get started with MicroPython on Raspberry Pi Pico by Gareth Halfacree and Ben Everard (book).
2. https://docs.micropython.org/en/latest/rp2/quickref.html
3. https://docs.micropython.org/en/latest/micropython-docs.pdf
4. https://www.coderdojotc.org/micropython/
5. https://www.raspberrypi.com/products/raspberry-pi-pico/
6. https://www.cytron.io/p-maker-pi-rp2040-simplifying-robotics-with-raspberry-pi-rp2040

Appendix A: Rev E

```
"""
Obstacle avoiding robot: Pico Bot
Adrie Huesman
Start: 1 Nov 2022
Last update: 21 Nov 2022
Hardware
2 120:1 Mini Plastic Gearmotors HP: https://www.pololu.com/product/1520
Pololu Mini Plastic Gearmotor Bracket Pair - Wide:
https://www.pololu.com/product/2680
Pololu Wheel 70×8mm Pair - Black: https://www.pololu.com/product/1425
Magnetic Encoder Pair Kit, 12 CPR, 2.7-18V:
https://www.pololu.com/product/1523
Swivel wheel: https://www.gamma.nl/assortiment/zwenkwiel-tpe-met-
plaatbevestiging-25mm-tot-15kg/p/B328710
Battery holder 4 AA
4 NiMH batteries AA, 12v and 2050 mAh
Maker Pi RP2040 board: https://www.cytron.io/p-maker-pi-rp2040-simplifying-
robotics-with-raspberry-pi-rp2040
270 degrees servo: https://www.bitsandparts.nl/Servo-motor-analoog-Micro-
Servo-9g-SG90-270%C2%B0-p1919329
Ultrasonic sensor RCLW-1601:
https://www.tinytronics.nl/shop/nl/sensoren/afstand/ultrasonische-sensor-
rcwl-1601
2 Pololu Distance Sensors with Pulse Width Output:
https://www.pololu.com/product/4064
Robot chassis 10 by 15 cm, two levels, made from 4 mm plywood:
https://quartel.nl/product/berken-triplex-25-x-100-4-0mm-1mtr-bet34-0/
Rev A Basic functionality.
Rev B Slow servo movement.
Rev C MFOspeed = f(USdistance).
Rev D Write data to file.
Rev E Use of machine.time_pulse_us for Ping, leftIR and rightIR.
"""

# Import libraries
import machine
import utime
import random
from neopixel import NeoPixel
utime.sleep(2) # for proper booting!

# Setup GPIO pins
# DC motors
M1A = machine.PWM(machine.Pin(8))
M1B = machine.PWM(machine.Pin(9))
M2A = machine.PWM(machine.Pin(10))
M2B = machine.PWM(machine.Pin(11))
M1A.freq(1000)
M1B.freq(1000)
M2A.freq(1000)
M2B.freq(1000)
# Servo
pwm1 = machine.PWM(machine.Pin(12))
pwm1.freq(50)
# US sensor
```

```
trigger = machine.Pin(17, machine.Pin.OUT) # send trigger
echo = machine.Pin(16, machine.Pin.IN) # get echo back
# Button
button1 = machine.Pin(20, machine.Pin.IN) # switch has external pull-up to
3.3v via 10 kohm
# Neopixel
pin = machine.Pin(18, machine.Pin.OUT)   # set GPIO18 to drive NeoPixels
np = NeoPixel(pin, 2) # create NeoPixel driver for 2 pixels
# Speaker
speaker = machine.PWM(machine.Pin(22))
# IR sensors
leftIRpin = machine.Pin(6, machine.Pin.IN)
rightIRpin = machine.Pin(26, machine.Pin.IN)

# Functions and interrupts
def leftmotor(speed):   # speed must be between -65535 and 65535
    if speed >= 0:
        M1A.duty_u16(0)
        M1B.duty_u16(speed)
    else:
        M1A.duty_u16(-speed)
        M1B.duty_u16(0)

def rightmotor(speed):   # speed must be between -65535 and 65535
    if speed >= 0:
        M2A.duty_u16(0)
        M2B.duty_u16(speed)
    else:
        M2A.duty_u16(-speed)
        M2B.duty_u16(0)

# Global values
leftsteps = 0
rightsteps = 0

# Interrupt Service Routine for left motor
def leftencoder(change):
    global leftsteps
    leftsteps += 1

# Define leftencoderA as being connected to GP6 and use the internal Pico
PULL_DOWN resistor
leftencoderA = machine.Pin(7, machine.Pin.IN) # encoder has external pull-
up to 3.3v via 10 kohm

# Associate the falling value on the input pin with the callback function
leftencoderA.irq(handler = leftencoder, trigger = machine.Pin.IRQ_FALLING)

# Interrupt Service Routine for right motor
def rightencoder(change):
    global rightsteps
    rightsteps += 1

# Define rightencoderA as being connected to GP2 and use the internal Pico
PULL_DOWN resistor
rightencoderA = machine.Pin(0, machine.Pin.IN) # encoder has external pull-
up to 3.3v via 10 kohm
```

```
# Associate the falling value on the input pin with the callback function
rightencoderA.irq(handler = rightencoder, trigger =
machine.Pin.IRQ_FALLING)

def move(direc, steps, aspeed):
    global leftsteps
    leftsteps = 0
    global rightsteps
    rightsteps= 0

    if direc == 'F':
        lfactor = 1
        rfactor = 1

    if direc == 'B':
        lfactor = -1
        rfactor = -1

    if direc == 'L':
        lfactor = -1
        rfactor = 1

    if direc == 'R':
        lfactor = 1
        rfactor = -1

    leftmotor(lfactor*aspeed)
    rightmotor(rfactor*aspeed)

    while leftsteps < steps: # and (rightsteps < steps):
        error = leftsteps - rightsteps
        paction = int(200.0*error)
        leftmotor(lfactor*(aspeed - paction))
        rightmotor(rfactor*(aspeed + paction))
        utime.sleep(0.005)

    leftmotor(0)
    rightmotor(0)

    #print('steps:', steps)
    #print('left:', leftsteps)
    #print('right:', rightsteps)
    #print('     ')

def ping():
    trigger.low()
    utime.sleep_us(5)
    trigger.high()
    utime.sleep_us(10)
    trigger.low()
    timepassed = machine.time_pulse_us(echo, 1, 14577) # time-out limits
distance to 250 cm
    if timepassed < 0:
        distance = 251
    else:
        distance = round((timepassed * 0.0343)/2, 1)
```

```
    return distance

def playtone(frequency):
    speaker.duty_u16(32768) # 50% dutycycle
    speaker.freq(frequency)
    utime.sleep(0.1)

def bequiet():
    speaker.duty_u16(0)

def leftIR():
    timepassed = machine.time_pulse_us(leftIRpin, 1, 35000) # time-out does
not limit distance!
    if timepassed < 999 or timepassed > 1999:
        distance = 301
    else:
        distance = round(0.4*(timepassed - 1000), 1)
    return distance # in cm

def rightIR():
    timepassed = machine.time_pulse_us(rightIRpin, 1, 35000) # time-out
does not limit distance!
    if timepassed < 999 or timepassed > 1999:
        distance = 301
    else:
        distance = round(0.4*(timepassed - 1000), 1)
    return distance # in cm

# Tuning
# Servo
sleft = 7195 # determined by servo2.py
scent = 5075
sright = 2850
# Limits
Climit = 30.0
RLlimit = 20.0
# MFO state
MFOspeed = 20000
# Neopixels
bright = 50 # max 255

# Main code
state = 'INI'
while True:
    if state == 'INI':
        np[0] = (bright, bright, bright)  # set right pixel to white
        np[1] = (bright, bright, bright)  # set left  pixel to white
        np.write()                        # write data to all pixels
        leftmotor(0)
        rightmotor(0)
        pwm1.duty_u16(scent)
        while button1.value():
            utime.sleep(0.1)
        utime.sleep(1.0)
        file = open("temps.txt", "w")
        state = 'SAM'
        np[0] = (0, 0, 0)  # set right pixel to off
```

```
    np[1] = (0, 0, 0)  # set left  pixel to off
    np.write()         # write data to all pixels

elif state == 'SAM':
    np[0] = (0, 0, bright)  # set right pixel to blue
    np[1] = (0, 0, bright)  # set left  pixel to blue
    np.write()              # write data to all pixels
    leftmotor(0)
    rightmotor(0)
    utime.sleep(1.0)
    USdistance = ping()
    IRleftdist = leftIR()
    IRrightdist = rightIR()
    if USdistance < Climit:
        Cprox = 1
    else:
        Cprox = 0
    if IRleftdist < RLlimit:
        Lprox = 1
    else:
        Lprox = 0
    if IRrightdist < RLlimit:
        Rprox = 1
    else:
        Rprox = 0
    if Cprox == 0 and Lprox == 0 and Rprox == 0:
        state = 'MFO'
    if Cprox == 1 and Lprox == 0 and Rprox == 0:
        state = 'T90'
    if Lprox == 1 and Rprox == 0:
        state = 'T15'
    if Lprox == 0 and Rprox == 1:
        state = 'T15'
    if Lprox == 1 and Rprox == 1:
        state = 'ESC'
    np[0] = (0, 0, 0)  # set right pixel to off
    np[1] = (0, 0, 0)  # set left  pixel to off
    np.write()         # write data to all pixels

elif state == 'T90':
    np[0] = (bright, 0, bright)  # set right pixel to magenta
    np[1] = (bright, 0, bright)  # set left  pixel to magenta
    np.write()                   # write data to all pixels
    for i in range(1, 6):
        freq = random.randint(300, 4000)
        playtone(freq)
    bequiet()
    for i in range(scent, (sleft + 1), 5):
        pwm1.duty_u16(i)
        utime.sleep_ms(1)
    utime.sleep(0.2)
    USldistance = ping()
    for i in range(sleft, (sright - 1), -5):
        pwm1.duty_u16(i)
        utime.sleep_ms(1)
    utime.sleep(0.2)
    USrdistance = ping()
```

```
    for i in range(sright, (scent + 1), 5):
        pwm1.duty_u16(i)
        utime.sleep_ms(1)
    utime.sleep(0.2)
    if USldistance > USrdistance:
        move('L', 174, 20000) # 174 steps should be 90 degrees
    else:
        move('R', 174, 20000) # 174 steps should be 90 degrees
    state = 'SAM'
    np[0] = (0, 0, 0)  # set right pixel to off
    np[1] = (0, 0, 0)  # set left  pixel to off
    np.write()         # write data to all pixels

elif state == 'MFO':
    np[0] = (0, bright, 0)  # set right pixel to green
    np[1] = (0, bright, 0)  # set left  pixel to green
    np.write()              # write data to all pixels
    leftsteps = 0
    rightsteps = 0
    lastmeas = 0
    leftmotor(MFOspeed)
    rightmotor(MFOspeed)
    while state == 'MFO':
        error = leftsteps - rightsteps
        paction = int(200.0*error)
        leftmotor(MFOspeed - paction)
        rightmotor(MFOspeed + paction)
        if (leftsteps - lastmeas) > 85: # take meas. each 5 cm
            lastmeas = leftsteps
            USdistance = ping()
            IRleftdist = leftIR()
            IRrightdist = rightIR()
            file.write(str(USdistance) + "\n")
            file.flush()
            if USdistance > 80.0:
                MFOspeed = int(1.05*MFOspeed)
                if MFOspeed > 32000:
                    MFOspeed = 32000
            if USdistance < 80.0:
                MFOspeed = int(0.95*MFOspeed)
                if MFOspeed < 20000:
                    MFOspeed = 20000
            if USdistance < Climit:
                Cprox = 1
            else:
                Cprox = 0
            if IRleftdist < RLlimit:
                Lprox = 1
            else:
                Lprox = 0
            if IRrightdist < RLlimit:
                Rprox = 1
            else:
                Rprox = 0
            if Cprox == 0 and Lprox == 0 and Rprox == 0:
                state = 'MFO'
            else:
```

```
                MFOspeed = 20000
                state = 'SAM'
    np[0] = (0, 0, 0)  # set right pixel to off
    np[1] = (0, 0, 0)  # set left  pixel to off
    np.write()         # write data to all pixels

elif state == 'T15':
    np[0] = (bright, bright, 0)  # set right pixel to yellow
    np[1] = (bright, bright, 0)  # set left  pixel to yellow
    np.write()                   # write data to all pixels
    if Lprox == 1:
        move('R', 29, 20000) # 174 steps should be 90 degrees
    else:
        move('L', 29, 20000) # 174 steps should be 90 degrees
    state = 'SAM'
    np[0] = (0, 0, 0)  # set right pixel to off
    np[1] = (0, 0, 0)  # set left  pixel to off
    np.write()         # write data to all pixels

elif state == 'ESC':
    np[0] = (bright, 0, 0)  # set right pixel to red
    np[1] = (bright, 0, 0)  # set left  pixel to re
    np.write()              # write data to all pixels
    move('B', 174, 20000)
    move('R', 348, 20000) # 174 steps should be 90 degrees
    state = 'SAM'
    np[0] = (0, 0, 0)  # set right pixel to off
    np[1] = (0, 0, 0)  # set left  pixel to off
    np.write()         # write data to all pixels
```

Appendix B: Rev J

```
"""
Obstacle avoiding robot: Pico Bot
Adrie Huesman
Start: 1 Nov 2022
Last update: 10 Dec 2022
Hardware
2 120:1 Mini Plastic Gearmotors HP: https://www.pololu.com/product/1520
Pololu Mini Plastic Gearmotor Bracket Pair - Wide:
https://www.pololu.com/product/2680
Pololu Wheel 70×8mm Pair - Black: https://www.pololu.com/product/1425
Magnetic Encoder Pair Kit, 12 CPR, 2.7-18V:
https://www.pololu.com/product/1523
Swivel wheel: https://www.gamma.nl/assortiment/zwenkwiel-tpe-met-
plaatbevestiging-25mm-tot-15kg/p/B328710
Battery holder 4 AA
4 NiMH batteries AA
Maker Pi RP2040 board: https://www.cytron.io/p-maker-pi-rp2040-simplifying-
robotics-with-raspberry-pi-rp2040
270 degrees servo: https://www.bitsandparts.nl/Servo-motor-analoog-Micro-
Servo-9g-SG90-270%C2%B0-p1919329
Ultrasonic sensor RCLW-1601:
https://www.tinytronics.nl/shop/nl/sensoren/afstand/ultrasonische-sensor-
rcwl-1601
2 Pololu Distance Sensors with Pulse Width Output:
https://www.pololu.com/product/4064
Robot chassis 10 by 15 cm, two levels, made from 4 mm plywood:
https://quartel.nl/product/berken-triplex-25-x-100-4-0mm-1mtr-bet34-0/
Rev A Basic functionality.
Rev B Slow servo movement.
Rev C MFO = f(USdistance).
Rev D Write data to file.
Rev E Use of machine.time_pulse_us for Ping, leftIR and rightIR.
Rev F Replaced US sensor/servo with VL53L1X sensor placed at the back of
the robot to increase FOV
Rev G Added US sensor at the back of the robot to double FOV, so there are
two sensors at the back
Rev H US sensor replaced by URM09 US sensor
Rev I VL53L1X and URM09 US sensor placed at the front
Rev J MFO was given a gradual stop at exit
"""

# Import libraries
import machine
import utime
import random
from neopixel import NeoPixel
from vl53l1x import VL53L1X # taken from
https://github.com/drakxtwo/vl53l1x_pico
utime.sleep(2) # for proper booting!

# Setup GPIO pins
# DC motors
M1A = machine.PWM(machine.Pin(8))
M1B = machine.PWM(machine.Pin(9))
M2A = machine.PWM(machine.Pin(10))
```

```
M2B = machine.PWM(machine.Pin(11))
M1A.freq(1000)
M1B.freq(1000)
M2A.freq(1000)
M2B.freq(1000)
# US sensor
trigger = machine.Pin(4, machine.Pin.OUT) # send trigger
echo = machine.Pin(5, machine.Pin.IN) # get echo back
# Button
button1 = machine.Pin(20, machine.Pin.IN) # switch has external pull-up to
3.3v via 10 kohm
# Neopixel
pin = machine.Pin(18, machine.Pin.OUT)   # set GPIO18 to drive NeoPixels
np = NeoPixel(pin, 2) # create NeoPixel driver for 2 pixels
# Speaker
speaker = machine.PWM(machine.Pin(22))
# IR sensors
leftIRpin = machine.Pin(6, machine.Pin.IN)
rightIRpin = machine.Pin(26, machine.Pin.IN)
# VL53L1X sensor
sda=machine.Pin(16)
scl=machine.Pin(17)
i2c=machine.I2C(0,sda=sda, scl=scl, freq=400000)
distance = VL53L1X(i2c)

# Functions and interrupts
def leftmotor(speed):   # speed must be between -65535 and 65535
    if speed >= 0:
        M1A.duty_u16(0)
        M1B.duty_u16(speed)
    else:
        M1A.duty_u16(-speed)
        M1B.duty_u16(0)

def rightmotor(speed):   # speed must be between -65535 and 65535
    if speed >= 0:
        M2A.duty_u16(0)
        M2B.duty_u16(speed)
    else:
        M2A.duty_u16(-speed)
        M2B.duty_u16(0)

# Global values
leftsteps = 0
rightsteps = 0

# Interrupt Service Routine for left motor
def leftencoder(change):
    global leftsteps
    leftsteps += 1

# Define leftencoderA as being connected to GP7
leftencoderA = machine.Pin(7, machine.Pin.IN) # encoder has external pull-
up to 3.3v via 10 kohm

# Associate the falling value on the input pin with the callback function
leftencoderA.irq(handler = leftencoder, trigger = machine.Pin.IRQ_FALLING)
```

```
# Interrupt Service Routine for right motor
def rightencoder(change):
    global rightsteps
    rightsteps += 1

# Define rightencoderA as being connected to GP0
rightencoderA = machine.Pin(0, machine.Pin.IN) # encoder has external pull-
up to 3.3v via 10 kohm

# Associate the falling value on the input pin with the callback function
rightencoderA.irq(handler = rightencoder, trigger =
machine.Pin.IRQ_FALLING)

def move(direc, steps, aspeed):
    global leftsteps
    leftsteps = 0
    global rightsteps
    rightsteps= 0

    if direc == 'F':
        lfactor = 1
        rfactor = 1

    if direc == 'B':
        lfactor = -1
        rfactor = -1

    if direc == 'L':
        lfactor = -1
        rfactor = 1

    if direc == 'R':
        lfactor = 1
        rfactor = -1

    leftmotor(lfactor*aspeed)
    rightmotor(rfactor*aspeed)

    while leftsteps < steps: # and (rightsteps < steps):
        error = leftsteps - rightsteps
        paction = int(200.0*error)
        leftmotor(lfactor*(aspeed - paction))
        rightmotor(rfactor*(aspeed + paction))
        utime.sleep(0.005)

    leftmotor(0)
    rightmotor(0)

    #print('steps:', steps)
    #print('left:', leftsteps)
    #print('right:', rightsteps)
    #print('     ')

def playtone(frequency):
    speaker.duty_u16(32768) # 50% dutycycle
    speaker.freq(frequency)
```

```
    utime.sleep(0.1)

def bequiet():
    speaker.duty_u16(0)

def leftIR():
    timepassed = machine.time_pulse_us(leftIRpin, 1, 35000) # time-out does
not limit distance!
    if timepassed < 999 or timepassed > 1999:
        distance = 301
    else:
        distance = round(0.4*(timepassed - 1000), 1)
    return distance # in cm

def rightIR():
    timepassed = machine.time_pulse_us(rightIRpin, 1, 35000) # time-out
does not limit distance!
    if timepassed < 999 or timepassed > 1999:
        distance = 301
    else:
        distance = round(0.4*(timepassed - 1000), 1)
    return distance # in cm

def ping():
    trigger = machine.Pin(5, machine.Pin.OUT)
    trigger.value(1)
    utime.sleep_us(10)
    trigger.value(0)
    trigger = machine.Pin(5, machine.Pin.IN)
    timepassed = machine.time_pulse_us(trigger, 1, 35000)
    if timepassed < 0:
        distance = 999
    else:
        distance = round((timepassed * 0.0343)/2, 1)
    return distance

# Tuning
# Limits
Climit = 30.0
RLlimit = 20.0
# MFO state
MFOspeed = 20000
# Neopixels
bright = 50 # max 255

# Main code
state = 'INI'
while True:
    if state == 'INI':
        np[0] = (bright, bright, bright)  # set right pixel to white
        np[1] = (bright, bright, bright)  # set left  pixel to white
        np.write()                        # write data to all pixels
        leftmotor(0)
        rightmotor(0)
        while button1.value():
            utime.sleep(0.1)
        utime.sleep(1.0)
```

```
    file = open("temps.txt", "w")
    state = 'SAM'
    np[0] = (0, 0, 0)  # set right pixel to off
    np[1] = (0, 0, 0)  # set left  pixel to off
    np.write()         # write data to all pixels

elif state == 'SAM':
    np[0] = (0, 0, bright)  # set right pixel to blue
    np[1] = (0, 0, bright)  # set left  pixel to blue
    np.write()              # write data to all pixels
    leftmotor(0)
    rightmotor(0)
    utime.sleep(1.0)
    IRdistance = (distance.read())/10
    USdistance = ping()
    centdistance = min(IRdistance, USdistance)
    IRleftdist = leftIR()
    IRrightdist = rightIR()
    if centdistance < Climit:
        Cprox = 1
    else:
        Cprox = 0
    if IRleftdist < RLlimit:
        Lprox = 1
    else:
        Lprox = 0
    if IRrightdist < RLlimit:
        Rprox = 1
    else:
        Rprox = 0
    if Cprox == 0 and Lprox == 0 and Rprox == 0:
        state = 'MFO'
    if Cprox == 1 and Lprox == 0 and Rprox == 0:
        state = 'T90'
    if Lprox == 1 and Rprox == 0:
        state = 'T15'
    if Lprox == 0 and Rprox == 1:
        state = 'T15'
    if Lprox == 1 and Rprox == 1:
        state = 'ESC'
    np[0] = (0, 0, 0)  # set right pixel to off
    np[1] = (0, 0, 0)  # set left  pixel to off
    np.write()         # write data to all pixels

elif state == 'T90':
    np[0] = (bright, 0, bright)  # set right pixel to magenta
    np[1] = (bright, 0, bright)  # set left  pixel to magenta
    np.write()                   # write data to all pixels
    for i in range(1, 6):
        freq = random.randint(300, 4000)
        playtone(freq)
    bequiet()
    USldistance = leftIR()
    USrdistance = rightIR()
    if USldistance > USrdistance:
        move('L', 174, 20000) # 174 steps should be 90 degrees
    else:
```

```
        move('R', 174, 20000) # 174 steps should be 90 degrees
    state = 'SAM'
    np[0] = (0, 0, 0)  # set right pixel to off
    np[1] = (0, 0, 0)  # set left  pixel to off
    np.write()         # write data to all pixels

elif state == 'MFO':
    np[0] = (0, bright, 0)  # set right pixel to green
    np[1] = (0, bright, 0)  # set left  pixel to green
    np.write()              # write data to all pixels
    leftsteps = 0
    rightsteps = 0
    lastmeas = 0
    leftmotor(MFOspeed)
    rightmotor(MFOspeed)
    while state == 'MFO':
        error = leftsteps - rightsteps
        paction = int(200.0*error)
        leftmotor(MFOspeed - paction)
        rightmotor(MFOspeed + paction)
        if (leftsteps - lastmeas) > 85: # take meas. each 5 cm
            lastmeas = leftsteps
            IRdistance = (distance.read())/10
            USdistance = ping()
            centdistance = min(IRdistance, USdistance)
            IRleftdist = leftIR()
            IRrightdist = rightIR()
            file.write(str(centdistance) + "\n")
            file.flush()
            if centdistance > 80.0:
                MFOspeed = int(1.05*MFOspeed)
                if MFOspeed > 32000:
                    MFOspeed = 32000
            if centdistance < 80.0:
                MFOspeed = int(0.95*MFOspeed)
                if MFOspeed < 20000:
                    MFOspeed = 20000
            if centdistance < Climit:
                Cprox = 1
            else:
                Cprox = 0
            if IRleftdist < RLlimit:
                Lprox = 1
            else:
                Lprox = 0
            if IRrightdist < RLlimit:
                Rprox = 1
            else:
                Rprox = 0
            if Cprox == 0 and Lprox == 0 and Rprox == 0:
                state = 'MFO'
            else:
                for i in range(1, 6):  # Do a gradual stop
                    MFOspeed = int(0.80*MFOspeed)
                    if MFOspeed < 10000:
                        MFOspeed = 10000
                    leftmotor(MFOspeed)
```

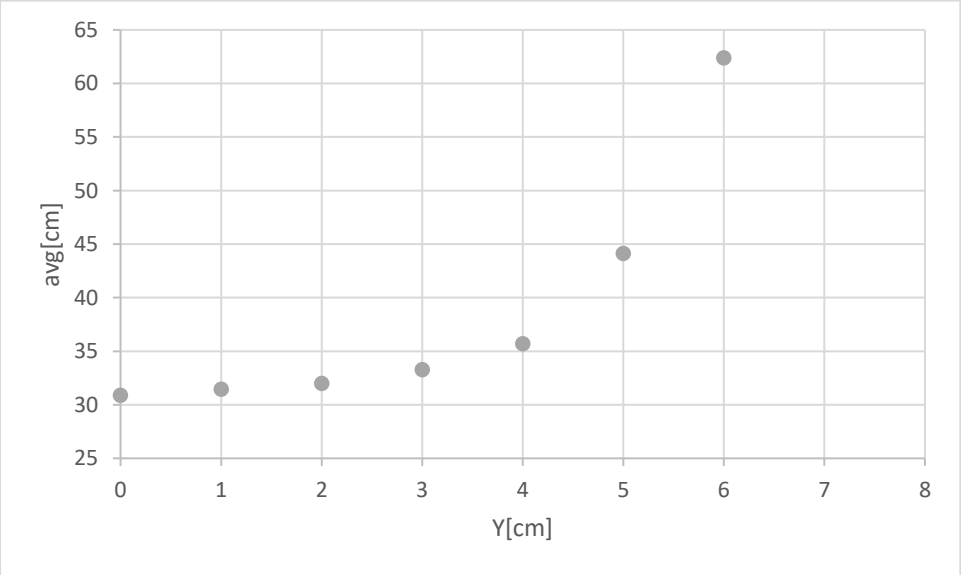
```
                    rightmotor(MFOspeed)
                    utime.sleep(0.1)
                leftmotor(0)
                rightmotor(0)
                MFOspeed = 20000
                state = 'SAM'
    np[0] = (0, 0, 0)  # set right pixel to off
    np[1] = (0, 0, 0)  # set left  pixel to off
    np.write()         # write data to all pixels

elif state == 'T15':
    np[0] = (bright, bright, 0)  # set right pixel to yellow
    np[1] = (bright, bright, 0)  # set left  pixel to yellow
    np.write()                   # write data to all pixels
    if Lprox == 1:
        move('R', 29, 20000) # 174 steps should be 90 degrees
    else:
        move('L', 29, 20000) # 174 steps should be 90 degrees
    state = 'SAM'
    np[0] = (0, 0, 0)  # set right pixel to off
    np[1] = (0, 0, 0)  # set left  pixel to off
    np.write()         # write data to all pixels

elif state == 'ESC':
    np[0] = (bright, 0, 0)  # set right pixel to red
    np[1] = (bright, 0, 0)  # set left  pixel to re
    np.write()              # write data to all pixels
    move('B', 174, 20000)
    move('R', 348, 20000) # 174 steps should be 90 degrees
    state = 'SAM'
    np[0] = (0, 0, 0)  # set right pixel to off
    np[1] = (0, 0, 0)  # set left  pixel to off
    np.write()         # write data to all pixels
```

Appendix C: Experimental FOV results

VL53L1X @ 30 cm				
Y[cm]	min [mm]	max [mm]	avg[cm]	Comments
0	307	311	30.9	
1	313	316	31.45	
2	318	322	32	
3	331	335	33.3	
4	354	360	35.7	
5	439	444	44.15	Below 45 cm
6	621	627	62.4	
7	674	681	67.75	
no object	678	682	68	
limit	5			
fov	18.92604			



URM09 @ 30 cm				
Y[cm]	min [cm]	max [cm]	avg[cm]	Comments
0	29.6	30	29.8	
1	30.1	30.1	30.1	
2	30.5	30.5	30.5	
3	30.2	30.2	30.2	
4	30.7	31.2	30.95	
5	32.1	32.5	32.3	
6	30.6	31.4	31	
7	32.9	33.7	33.3	
8	31.5	32.3	31.9	
9	35.5	37.1	36.3	Widening!
10	33.7	37.9	35.8	
no object	46.2	49.6	47.9	
limit	8			
fov	29.86503