THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

DEPARTMENT OF
COMPUTER SCIENCE
Te Tari Rorohiko

2024

Contributors

J. Turner

V. Moxham-Bettridge

TRAINING BERTIE

Last week we introduced you to Bertie and the basic movements available. Today you will get to experiment with Petoi's extensible modules to teach Bertie obedience skills in preparation for The Dog Show event in the next session.

Extensible Modules

As mentioned in the last session, Petoi make modules that extend the robot's capabilities (see figure 14). These include a light sensor to measure brightness, a touch sensor to detect physical contact, a gesture sensor to recognise simple hand gestures, a PIR motion sensor to detect movement of a warm object and an IR distance sensor to measure the distance between the sensors and objects located in front of the sensors.

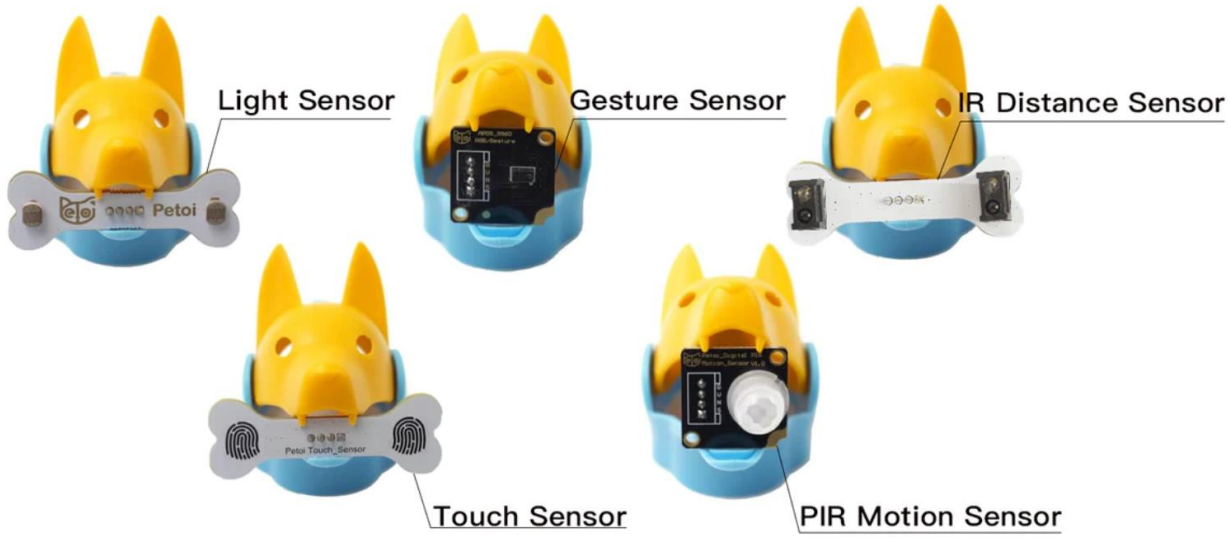

Figure 14: The Sensors Included in the Petoi Basic Sensor Pack

IR Distance Sensor

If you did the line following session using Micro:bits and the Maqueen robot, you may already be familiar with the concept of IR sensors. However, today we use them for more than just detecting a black line on a light surface, today we're going to explore how they detect distance.

An Active IR sensor consists of an IR transmitter and an IR receiver in a single unit (see figure 15). The IR transmitter emits IR light which is then reflected by objects in front of the sensor and this reflection is detected by the IR receiver. The distance can then be determined by the intensity of the reflection.

Figure 15: The IR Distance Sensor

Firstly, to use the sensor, it must be connected. If the sensor is not already connected, follow these steps (the video demonstrates these in better detail):

1. Get out the IR Distance sensor and cable so they are ready.
2. Turn Bertie OFF.
3. Remove the back cover.
4. Carefully thread the cable through Bertie's mouth.
5. Plug one end of the cable into the **analogue pins socket** (labelled A2 & A3).
6. Plug the other end into the IR Distance sensor socket.
7. Pull the cable through enough so that the sensor will sit in the mouth without moving when the mouth is shut.
8. Turn Bertie ON.

Now the sensor is ready to be used! If looking at the robot head on, the left IR sensor uses the A2 pin and the right IR sensor uses the A3 pin.

Firstly, to see what kind of values the sensors output, write a few print statements as in figure 16 below and run the code. What happens if you hold the white box in front of the sensors? What happens if you move it closer and then further away? How do the values change?

```
1   from PetoiRobot import *
2
3   openPort("COM6")
4
5   while True:
6       # read the values of each IR sensor
7       # and store them in variables
8       a2 = readAnalogValue(2)
9       a3 = readAnalogValue(3)
10
11      # see the value of each sensor
12      print("A2: " + str(a2))
13      print("A3: " + str(a3))
```

Figure 16: Code for Reading the IR Sensors' Values

Initially (without anything in front of the sensors), the values would've been high (in the 900+ range), but they should have started to decrease once the white box was within 60cm (approx.) of the sensors. As the white box gets closer to the sensors, the values should decrease, and as the white box gets further away, the values should increase.

Knowing this, how would you program Norbert to sit whenever the white box approaches (i.e. within 15cm approx.) and then to walk when the white box is not near? Attempt this now.

It's okay if your code looks slightly different to what is shown in figure 17 because the values which deem the box too close or too far depend on various factors, such as ambient lighting and the other objects in the room.

```
1   from PetoiRobot import *
2
3   openPort("COM6")
4
5   while True:
6       # read the values of each IR sensor
7       # and store them in variables
8       a2 = readAnalogValue(2)
9       a3 = readAnalogValue(3)
10
11      # see the value of each sensor
12      print("A2: " + str(a2))
13      print("A3: " + str(a3))
14
15      if a2 < 60 and a3 < 60:
16          # both sensors read less than 60
17          # the box is close enough so Norbert
18          # should sit.
19          sendSkillStr('ksit', 0.5)
20      else:
21          # both sensors read more than or equal to 60
22          # so the box is NOT close enough. Norbert
23          # should walk forward.
24          sendSkillStr('kwkF', 0.5)
```

Figure 17: Code for Instructing Norbert to Walk or Sit Based on the Readings from the IR Sensors

Touch Sensor

How the Touch Sensor works is based on the principle of electrical capacitance (a system's ability to store electric charge). The sensor's contact patches (see figure 18) are constantly supplied with an AC signal which generates an electric field. When a conductive material (such as a finger) approaches the contact patches, the electric field is disrupted, thus changing the capacitance. This signals to the sensor's controller that something has come into contact with the sensor (ElectricalHub, 2024).

Figure 18: The Touch Sensor

The process for connecting this sensor is as follows:

1. Turn Norbert OFF.
2. Unplug the Bluetooth adapter.
3. Carefully unplug the cable from the analogue pin socket.
4. Plug the cable into a **digital pin socket** (either labelled D6 & D7 or D8 & D9).
5. Disconnect the IR Distance sensor.
6. Connect the Touch sensor.
7. Plug the Bluetooth adapter back in.
8. Turn Norbert ON.

The touch sensor is easy to program, as if a contact patch is pressed, it returns 1, otherwise it returns 0. If looking at the robot head on, the left contact patch is connected to digital pin D6 (or D8) and the right contact patch is connected to digital pin D7 (or D9).

Knowing this, let's program Norbert to turn his head when the sensor is pressed. If the left contact patch is pressed, Norbert should turn his head to the left, and if the right one is pressed, Norbert should turn his head to the right. If both are pressed, Norbert's head should return to the centre. See the code below (figure 19).

```
from PetoiRobot import *

openPort("COM6")

while True:
    # read the values of the contact
    # patches and store them in variables
    d6 = readDigitalValue(6)
    d7 = readDigitalValue(7)

    # see the value of each contact patch
    print("D6: " + str(d6))
    print("D7: " + str(d7))

    if d6 == 1 and d7 == 1:
        # both contact patches are pressed
        # move head to centre (0 degree angle)
        rotateJoints('M', absValList(0, 0), 1)
    elif d6 == 1:
        # left contact patch is pressed
        # move head to the left (-30 degrees angle)
        rotateJoints('M', absValList(0, -30), 1)
    elif d7 == 1:
        # right contact patch is pressed
        # move head to the right (30 degrees angle)
        rotateJoints('M', absValList(0, 30), 1)
```

Figure 19: Code for Moving Norbert's Head Using the Touch Sensor

PIR Motion Sensor

You may not be as familiar with this type of sensor (see figure 20) as much as the others we have looked at previously but it is likely that you have encountered them in some form similar to that of the security light shown in figure 21.

Figure 20: PIR Motion Sensor

Figure 21: Common Security Light with an Integrated PIR Motion Sensor
Retrieved from the 'Movement Sensors: How do they work?' article (link in 'Useful Resources' section)

PIR stands for 'Passive Infrared' and works by detecting changes in heat energy in a particular environment. When something warmer than the ambient temperature enters the scope of the sensor, provided the difference is greater than the sensor's threshold (i.e. how big of a difference before the sensor recognises it as movement), it will be detected by the sensor. This works because basically everything (people, animals and inanimate objects) emits IR radiation so when you walk in front of the sensor, provided you are warmer than the ambient temperature of the room, you will emit more IR radiation which will be detected by the sensor and recognised as movement. This is why PIR sensors are commonly used in home security systems because alerting there has been movement is sufficient (rather than needing to know the exact location of the movement).

The PIR sensor also uses digital pins like the Touch sensor so the process to connect the sensor is quick and easy:

1. Turn Norbert OFF.
2. Disconnect Touch sensor.
3. Connect PIR Motion sensor (see figure 14 for orientation).
4. Turn Norbert ON.

Unlike the other 2 sensors used in this session, the PIR Motion sensor only uses one digital pin, D6 (or D8). The easiest way to see how the sensor works in practice is to use an onboard LED to show the presence of movement (see figure 22 below).

```
1   from PetoiRobot import *
2
3   openPort("COM6")
4
5   while True:
6       # get the value of the PIR sensor
7       d6 = readDigitalValue(6);
8
9       if d6 == 1:
10          # movement has been detected so turn LED ON
11          writeDigitalValue(3, 1)
12      elif d6 == 0:
13          # no movement detected so turn LED OFF
14          writeDigitalValue(3, 0)
15
```

Figure 22: Basic Code that Turns an LED On if Movement is Detected

The `writeDigitalValue` function takes 2 values: the pin to change the value of (e.g. 3 corresponds to `LED3` on the circuit board) and the value to change it to (1 for `HIGH` and 0 for `LOW`).

To see how the sensor works, place Bertie approximately 1.5m - 2m from a wall (with the sensor facing the wall) and then walk past Bertie with at least 1m between you and the robot (see figures 23 & 24). What happens? Does the LED light up? How long does the sensor take to

register your movement? How long does the sensor take to realise you have gone? Feel free to ask staff for assistance if the instructions are unclear.

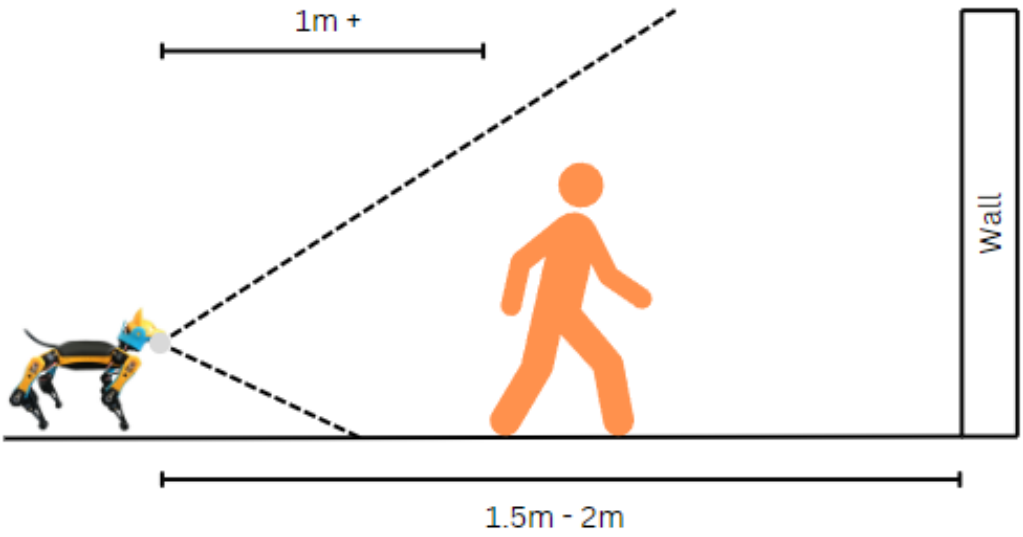

Figure 23: How to Position Bertie with the PIR Motion Sensor Attached (Side View)

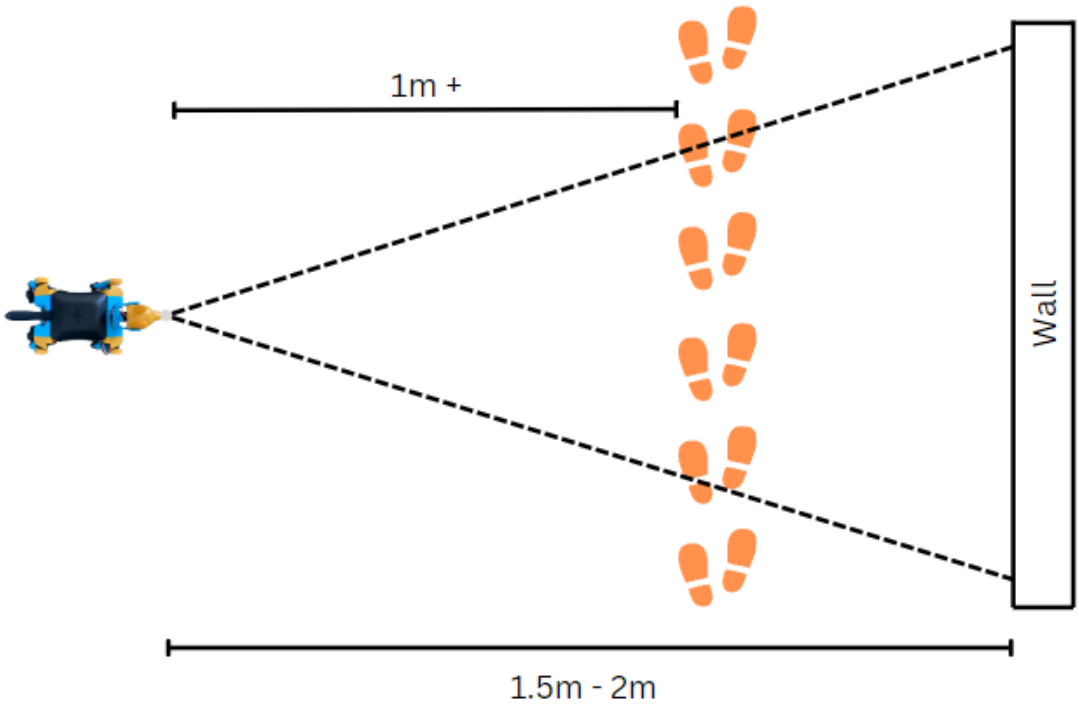

Figure 24: How to Position Bertie with the PIR Motion Sensor Attached (Top View)

When you walked in front of Bertie, a green LED should have turned on and then once past Bertie, the green LED should have turned off. Did you take note how long it took for each action to register with the sensor? Hopefully you noticed that when you walked in front of Bertie, it

detected your presence quickly (i.e. within a second) but once you had walked away, there was a delay in the LED turning off (i.e. taking anywhere from 2 – 8 seconds before turning off).

This is likely due to the duration of which the sensor retains the movement detected value (i.e. how long it returns 1). Often a sensor's behaviour can be tweaked, however as there are no Python commands included by Petoi to do this, the delay experienced is an example of a hardware limitation. It doesn't matter how we program the sensor with Python, as there will always be a lag in the sensor recognising the absence of movement.

Today's Exercise

In the last session, you taught Bertie basic agility skills so in today's session you will teach Bertie basic obedience skills. What Bertie needs to do is follow the white box until the box crosses the road, at which point Bertie must stop and look left, right and then left before crossing the road and following the box again (remember to save your code in a safe place as you may find it helpful for The Dog Show!).

Exercises:

1. Connect the IR Distance sensor using the steps outlined earlier in the session.
2. Bertie can start in any position (i.e. sitting, standing or resting).
3. When the white box is within 30cm of Bertie, Bertie should follow the box (up to you if this is a crawl, walk or trot).
4. Once Bertie is close to the box (i.e. within 20cm or so), Bertie should stop (again, up to you what action Bertie does – sit, stand, rest, step etc.)
5. Steps 3 & 4 can happen any number of times so make sure to use an infinite loop.
6. At some point Bertie will come to a road which they need to look left, right and then left again before crossing. This can be done by:
    a. When the box is far from Bertie (i.e. further than 30cm away), turn Bertie's head to the left, then right and then left again.
    b. Instruct Bertie to walk forward until step 3 or 4 occurs.

Like the last session, once you think Bertie is trained up, ask a staff member to watch your performance!

Summary

In this session, we taught Bertie basic obedience skills by using a few of the Extensible Modules made by Petoi. We discussed the theory behind the IR Distance, Touch and PIR Motion sensors and programmed them to make Bertie detect obstacles, touch and movement. We discovered that the PIR Motion sensor is an example of a hardware limitation because regardless of how we program the sensor, there will always be a delay in the sensor recognising the movement has

stopped. Now that sessions 1 & 2 are complete, it means the next session is The Dog Show where you showcase all the hard work you've put into training Bertie!

Useful Resources

- IR Sensor Working: https://robocraze.com/blogs/post/ir-sensor-working
- ElectricalHub (2024, Jan) *Touch Sensors: A Comprehensive Guide*. https://www.linkedin.com/pulse/touch-sensors-comprehensive-guide-electrical-hub-v3yqf?trk=article-ssr-frontend-pulse_more-articles_related-content-card#:~:text=The%20touch%20sensor's%20surface%20is,it%20disrupts%20the%20electrostatic%20field.
- How Do Touch Sensitive Screens Work?: https://engineering.mit.edu/engage/ask-an-engineer/how-do-touch-sensitive-screens-work/
- Movement Sensors: How do they work?: https://www.kennerelectrics.com.au/blog/movement-sensors-how-do-they-work#:~:text=Passive%20Infrared%20(PIR)%20Movement%20Sensors,are%20warmer%20than%20their%20surroundings.
- The Right Tool for the Job: Active and Passive Infrared Sensors: https://www.arrow.com/en/research-and-events/articles/understanding-active-and-passive-infrared-sensors
- Infrared Sensor: Types, Working Principle, and Applications (this article includes a helpful diagram demonstrating the difference between passive and active IR sensors): https://www.easybom.com/blog/a/infrared-sensor-types-working-principle-and-applications