Lesson 17 – Project – Bike Collision Detection

Setting the Scene

Riding a bike is very popular both for pleasure and also as a mode of transport to get to school or work. Staying safe is important and you have been approached by a tech company to design, build test and demonstrate a bike collision detection system. The system should be attached to the bike and then relay messages to the rider about their balance or stability.

## Success Criteria

* The micro:bit responds to acceleration on one of the axes, x y or z
* The micro:bit responds to acceleration on a combination of **two** of the axes, x y or z
* The micro:bit responds to acceleration on all three axes
* The responses are standard, LED or scrolling text message
* The response uses the music or speech warnings
* The values used are suitable and tested.
* You stay safe on your bike

**Some Ideas**

Here are some possible ideas that could be programmed:

* If the rider leans too far to the left, then the system flashes or plays an audio warning
* If the rider performs a wheelie then the system provides a warning message.
* The system provides different levels of warning,
* The system provides feedback on braking and if the rider is braking safely

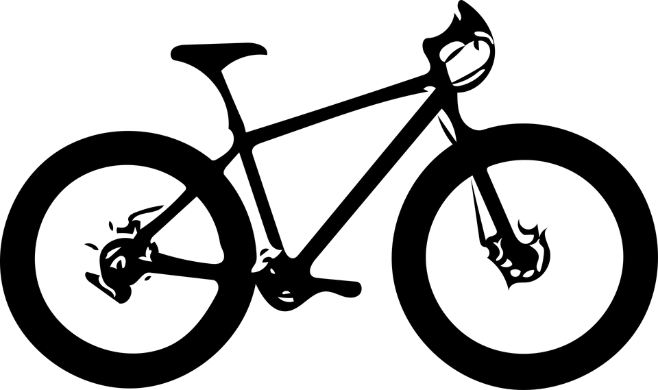
**Design**

Design and build a container for the micro:bit so that it can attach to the bike. You will need to consider where on the bike the micro:bit will be attached as this will affect the readings and values required for measuring the acceleration.

1. The system could be attached the handlebars
2. The system could be attached to the crossbar
3. Or the A frame

1

2



3

The container needs to be robust enough to hold and protect the micro:bit as well as coping with the demands of riding the bike at speed and braking. **The system should not interfere with the ability to ride the bike safely**.

## **Materials Needed**

## Think about the materials that you need to create this. A method to protect the micro:bit system and also attach it securely to the bike. It needs to be able to withstand the normal actions of riding a bike.

## Getting Started

The example program shows you how to measure and respond to acceleration along the *x*-axis. This measures the micro:bit being tilted to either the left or the right.

from microbit import \*

while True:

reading = accelerometer.get\_x()

if reading > 20:

display.show("R")

elif reading < -20:

display.show("L")

else:

display.show("-")

The *y*-axis relates to the micro:bit being moved up and down. Doing a wheelie tilts the bike up or pressing the front brake only tilts the bike downward. Adapt the program to measure these movements. Remember that the value 20 can be adjusted.

from microbit import \*

while True:

reading = accelerometer.get\_x()

if reading > 20:

display.show("R")

elif reading < -20:

display.show("L")

else:

display.show("-")

Now adapt your program to measure and respond to the z-axis. Remember this the forward and backward movement. Again you may need to adjust the values as the bike will accelerate forward as you start cycle or speed up. However, the bike should not be moving backwards!

## **Building a crash detector for a bike**

We can use the acceleration to measure a change in direction. Imagine that you are riding your bike in a straight line and then you start to turn a left-hand corner. As you lean to the left, this changes the speed or the direction or both so the acceleration changes. If you micro:bit was attached to your bike it would flash up the letter ‘L’.

Adapt the program below and then download to your micro:bit. The you can test out the various movements and change the values to get realistic readings and responses.

# Add your Python code here. E.g.

from microbit import \*

while True:

readingx = accelerometer.get\_x()

readingy = accelerometer.get\_y()

Create additional variables to store the y axis readings.

if readingx > 20:

display.show("R")

elif readingx < -20:

display.show("L")

Here we add a further set of elif statements to respond to the *y*-axis acceleration readings. These are combined with the *x*-axis readings so that we can measure changes across the axis.

elif readingy > 20:

display.show("U")

elif readingy < -20:

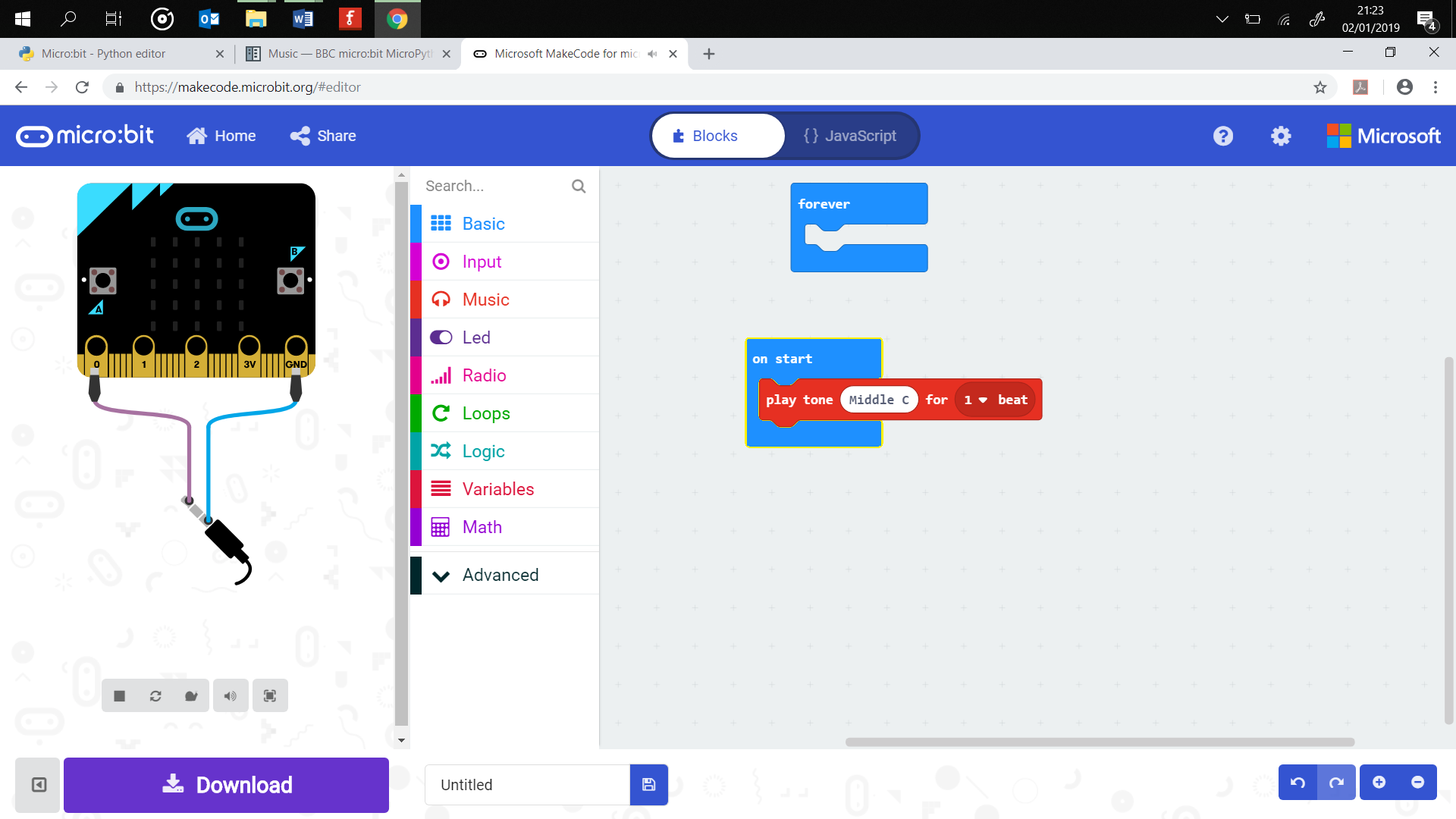
display.show("D")

else:

display.show("-")

## Pro-tip

If you are adding a music or speech warning to your project, then you will need to wire up a speaker as shown in the diagram below.



Use or adapt the code below to add a sound response to your crash detector.

from microbit import \*

import music

music.play(music.NYAN)

## Test Time

**Testing needs to be conducted safely**. You can simulate the bike by mounting the micro:bit to a pole. This can be used to represent the handlebars or part of the frame. You can then move the pole in the required positions to test the program.

You will need to adapt the values in order to get usable readings, for example reading < -20:maybe too large and you have already fallen off your bike!

Another option is to try a real bike. Attach the micro:bit to the frame or handle bars and try leaning side to side. **Remember to carry out all testing in a safe environment.**

## Stretch Tasks

* You add to respond to acceleration on 2 axes
* You add to respond to acceleration on all 3 axes
* Build in an earlier warning system that alerts the riding if they are going off balance. This can be used to prevent the crash

## Final Thoughts

This has been a program that has combined a lot of the elements from the previous lessons. You have also had to test your program and adjust the values to create responses that work in the real world. It is not good the bike crash detector triggering every time you brake!