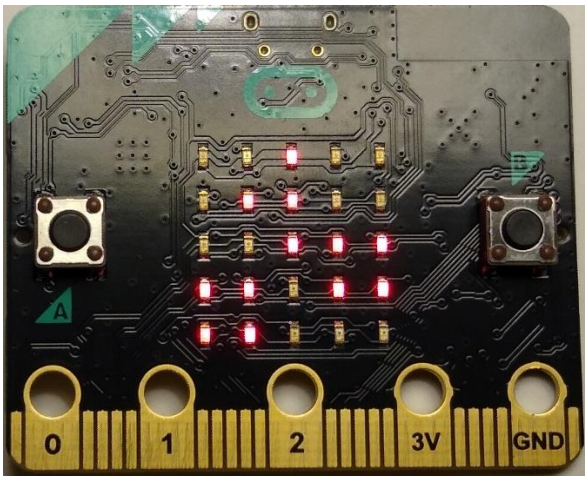
Wheelieometer for the Micro:Bit



A simple wheelieometer code has been written for the Micro:Bit to time wheelies.

What you need

- A BBC Micro:Bit
- The program code (hex file) to load on to the Micro:Bit
- A battery pack for the Micro:Bit to power it when attached to the bike
- Elastic bands or similar to attach the Micro:Bit to the handlebars
- (optional) A case to protect the Micro:Bit
- (optional) Two LEDs , two 1 kΩ resistors, three 4 mm plugs and some wire
- (optional) A small piezo-electric loudspeaker, wire and 4 mm plug (v.2 of the Micro:Bit already has a built-in speaker)

Instructions

- Copy the hex file to the Micro:Bit, once loaded the Micro:Bit will reboot and the file will disappear.
- Mount the Micro:Bit on the handlebars so it is somewhere between horizontal (flat) and vertical, with the LED screen facing you.
- Make sure the bike is on level ground and not leaning to the side; the program takes a calibration reading o account for the mounting angle when the Micro:Bit is first switched on.
- Connect the battery and turn the Micro:Bit on. The screen will show a symbol of a bike on the level.
- Start wheelieing. The screen will show a symbol of a bike wheelieing.
- At the end of the wheelie, the screen will return to a symbol of a bike on the level.
- Press Button A to display the length of the last wheelie in seconds. If you have set a new record, the time will be followed by *(Record)*.
- Press Button B to display the length of the record wheelie in seconds.
- If you want to clear the record wheelie time, press the Reset button (between the USB and battery connectors)

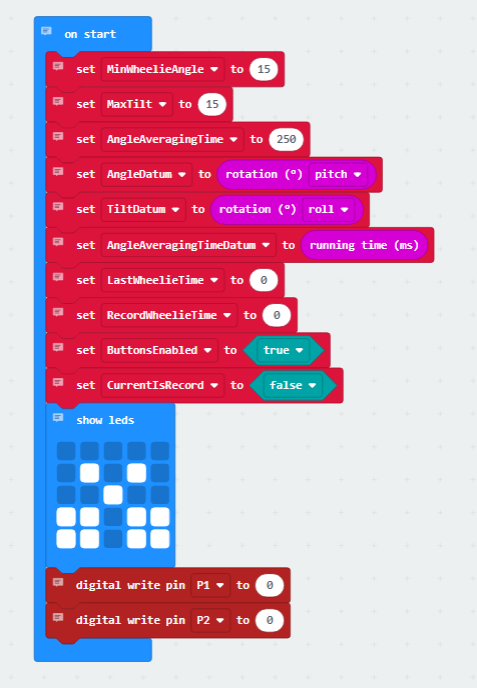
Optional Components

- The start and end of a wheelie are indicated by a high and low beep respectively, and a new record by a two tone beep. This requires a piezo-electric loudspeaker wired between Pin 0 and Ground on the version 1 Micro:Bit (Version 2 already has a built-in speaker).
- Wheelie in progress and a new record can be indicated on LEDs wired between Pin 1 and ground, and between Pin 2 and ground. Connect the anode leg of the LED to pin 1 or 2 via a 1 kΩ resistor, and the cathode leg to Ground. The anode leg is longer and the body of the LED is flattened on the side of the cathode leg.

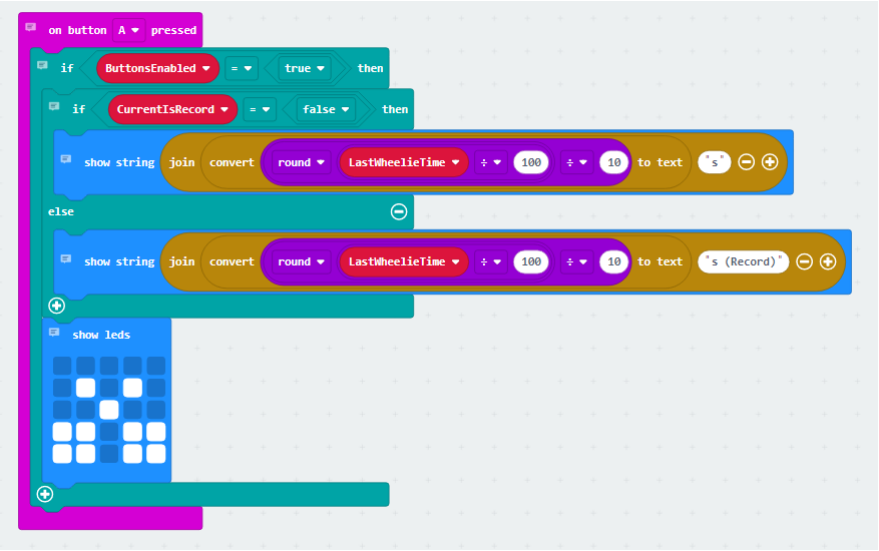
Code Listing

Block View

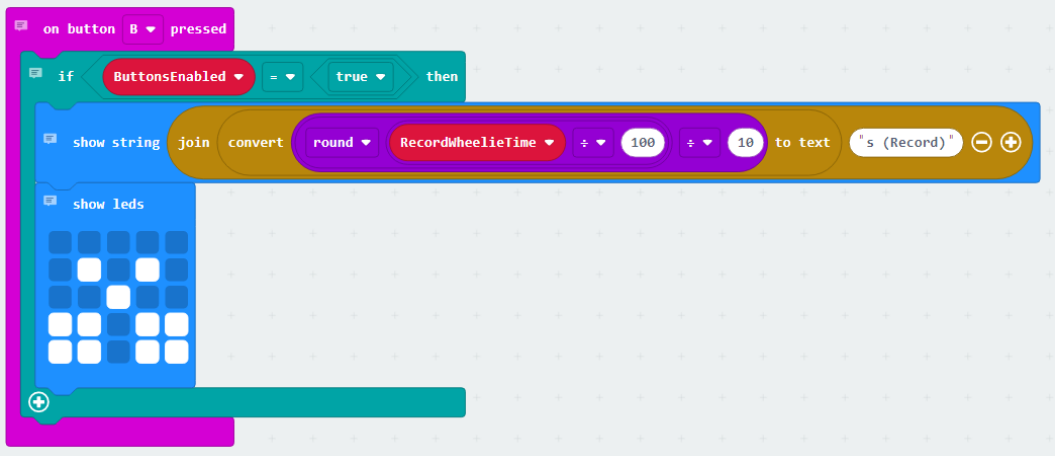
Initialisation



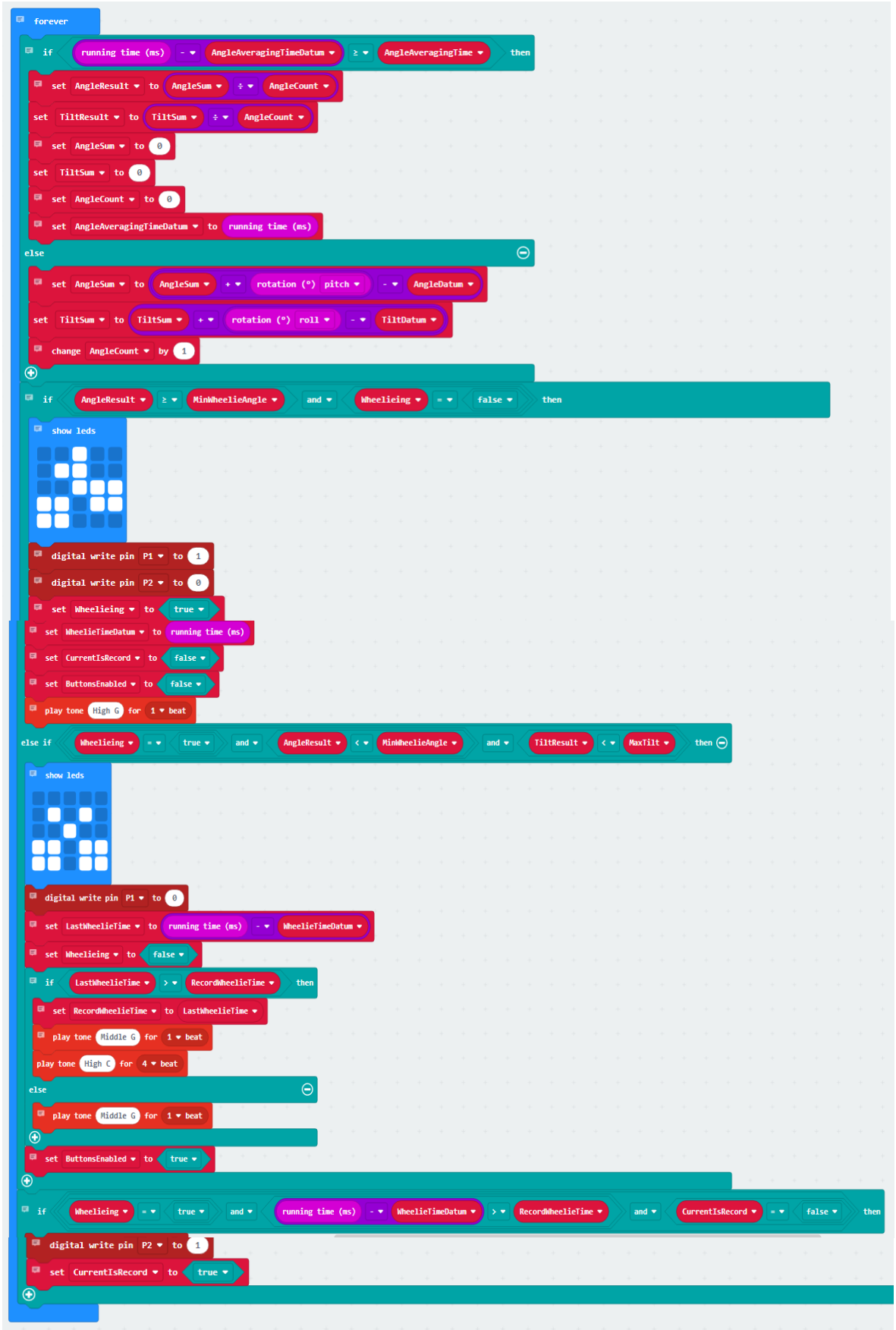
Button A



Button B



Main Loop



JavaScript

```
// Show latest recorded time rounded to 1 decimal place
input.onButtonPressed(Button.A, function () {
    // Button disabled during wheelie
    if (ButtonsEnabled == true) {
        // If last wheelie was a new record, this is indicated
        if (CurrentIsRecord == false) {
            // Show last wheelie time, rounded to one decimal place
```

```
            basic.showString("" + convertToText(Math.round(LastWheelieTime / 100) / 10) + "s")
        } else {
            // Indicate that the current time is a new record time
            basic.showString("" + convertToText(Math.round(LastWheelieTime / 100) / 10) + "s (Record)")
        }
        // Return to normal display
        basic.showLeds(`
            . . . . .
            . # . # .
            . . # . .
            # # . # #
            # # . # #
            `)
    }
})
// Show record wheelie time rounded to 1 decimal place
input.onButtonPressed(Button.B, function () {
    // Buttons are locked during wheelie
    if (ButtonsEnabled == true) {
        // Show record time rounded to one decimal place
        basic.showString("" + convertToText(Math.round(RecordWheelieTime / 100) / 10) + "s (Record)")
        // Return to normal display
        basic.showLeds(`
            . . . . .
            . # . # .
            . . # . .
            # # . # #
            # # . # #
            `)
    }
})
// Initialisation
let WheelieTimeDatum = 0
let Wheelieing = false
let TiltSum = 0
let TiltResult = 0
let AngleCount = 0
let AngleSum = 0
let AngleResult = 0
let CurrentIsRecord = false
let ButtonsEnabled = false
let RecordWheelieTime = 0
let LastWheelieTime = 0
// Minimum front/back angle required to detect wheelie
let MinWheelieAngle = 15
// Maximum allowable  sideways tilt  at end of wheelie (stops end of wheelie being detected early if rider steers in the
air)
let MaxTilt = 15
// Time in milliseconds to average angle measurements over
let AngleAveragingTime = 250
// Measured bike front/back angle when level - compensates for mounting angle
let AngleDatum = input.rotation(Rotation.Pitch)
// Measured bike sideways tilt angle when level - compensates for mounting angle
let TiltDatum = input.rotation(Rotation.Roll)
// Timer value at start of an angle averaging period
let AngleAveragingTimeDatum = input.runningTime()
// Length of last measured wheelie in milliseconds
LastWheelieTime = 0
// Length of longest measured wheelie since startup in milliseconds
RecordWheelieTime = 0
// Whether A and B buttons enabled (buttons locked during wheelie)
ButtonsEnabled = true
// Whether last wheelie was a new record
CurrentIsRecord = false
// Initialise screen
basic.showLeds(`
    . . . . .
    . # . # .
    . . # . .
    # # . # #
    # # . # #
    `)
// Initialise Pin 1 output for wheelie in progress
pins.digitalWritePin(DigitalPin.P1, 0)
// Initialise pin 2 output for last wheelie was a new record
pins.digitalWritePin(DigitalPin.P2, 0)
// Main loop
basic.forever(function () {
    // Average angle calculation
    if (input.runningTime() - AngleAveragingTimeDatum >= AngleAveragingTime) {
        // Calculate result at end of averaging time, divide summed values by number of readings
        AngleResult = AngleSum / AngleCount
        TiltResult = TiltSum / AngleCount
        // Reset summations for new calculation
        AngleSum = 0
        TiltSum = 0
        // Reset number of readings
        AngleCount = 0
        // Reset timer value at start of averaging period
        AngleAveragingTimeDatum = input.runningTime()
    } else {
        // During the averaging period, sum the readings, subtract datum angle each time to compensate for mounting angle
        AngleSum = AngleSum + input.rotation(Rotation.Pitch) - AngleDatum
        TiltSum = TiltSum + (input.rotation(Rotation.Roll) - TiltDatum)
        // Increment count of number of readings
        AngleCount += 1
    }
```

```
    // Detect wheelie start and end
    if (AngleResult >= MinWheelieAngle && Wheelieing == false) {
        // New wheelie detected, update display
        basic.showLeds(`
            . . # . .
            . # # . .
            . . # # #
            # # . # #
            # # . . .
            `)
        // Output pin 1 high for Wheelie in Progress indication
        pins.digitalWritePin(DigitalPin.P1, 1)
        // Output Pin 2 low, current wheelie not a new record
        pins.digitalWritePin(DigitalPin.P2, 0)
        // Wheelie is in progress
        Wheelieing = true
        // Record timer value at start of wheelie
        WheelieTimeDatum = input.runningTime()
        // Current wheelie is not yet a new record
        CurrentIsRecord = false
        // Lock buttons
        ButtonsEnabled = false
        // Start of wheelie sound
        music.playTone(784, music.beat(BeatFraction.Whole))
    } else if (Wheelieing == true && AngleResult < MinWheelieAngle && TiltResult < MaxTilt) {
        // Return to default display
        basic.showLeds(`
            . . . . .
            . # . # .
            . . # . .
            # # . # #
            # # . # #
            `)
        // Pin 1 low as wheelie not in progress
        pins.digitalWritePin(DigitalPin.P1, 0)
        // Calculate length of wheelie
        LastWheelieTime = input.runningTime() - WheelieTimeDatum
        // Note wheelie not in progress
        Wheelieing = false
        // Check if a new record has been set
        if (LastWheelieTime > RecordWheelieTime) {
            // Store new record time
            RecordWheelieTime = LastWheelieTime
            // Play special sound for new record
            music.playTone(392, music.beat(BeatFraction.Whole))
            music.playTone(523, music.beat(BeatFraction.Breve))
        } else {
            // Play end of wheelie sound
            music.playTone(392, music.beat(BeatFraction.Whole))
        }
        // Unlock buttons
        ButtonsEnabled = true
    }
    // Detect a new record as soon as previous record exceeded
    if (Wheelieing == true && input.runningTime() - WheelieTimeDatum > RecordWheelieTime && CurrentIsRecord == false) {
        // Set pin 2 high as soon as length of wheelie exceeds current record
        pins.digitalWritePin(DigitalPin.P2, 1)
        // Note that a new record has been achieved
        CurrentIsRecord = true
    }
})
```