Section 1: Digital and Analog pins for sensing and control

Throughout EPROM we will be using the Microsoft MakeCode JavaScript blocks editor. To illustrate some of the key hardware ideas I will use the ‘Electronics Starter Kit for Micro:bit’ from Monk Makes at £30. There is a [YouTube video here](https://www.youtube.com/watch?v=example_video_id). The three output devices are a buzzer, a lamp and a motor. Each can be connected directly to the power source, a single AA 1.5V battery. By making and breaking the circuit by connecting and disconnecting one of the alligator clips you can have simple physical control. So you could send Morse code messages via the buzzer or by flashing the light. There are also three external sensors on the Sensor Board: light, temperature and sound.

The photo shows the kit, together with the instruction booklet and a BBC micro:bit. I have wired up a very simple circuit consisting just of a single AA 1.5V battery, a light bulb and a pair of leads with crocodile clips. There isn’t even a switch! You can just connect and disconnect one end of one of the leads to make or break the circuit. You could make a Morse code tapper by just using the black lead to tap the negative screw on the bulb holder. But as we have a micro:bit to hand, let’s explore how that can be used to control the powered circuit.

The key device in connecting electrical and electronic components together is called a ‘relay’. There is a ‘Relay Board’ just above the red lead. More information is [here](https://example.com/relay-board). It can be purchased separately for about £6.60. An alternative to a relay is to use a transistor. This effectively adds a switch
to the circuit, as shown. In order to make the micro:bit do the switching we can connect the GND terminal of the relay board to the GND pin of the micro:bit. Then we connect any of the micro:bit’s I/O (input and/or output) pins to the IN terminal of the relay board. The simplest program to control the circuit just use button A to switch the bulb on, and button B to switch it off. We just need to write a 0 or 1 to chosen pin. Here it is in the MakeCode editor. I have chosen to use the P2 pin, and the simulator shows the result of pressing button A. The ‘On button’ block comes from the ‘Input’ menu. To get the ‘digital write’ block you need to first select ‘Advanced’ and then find the ‘Pins’ menu.

Remember to give your program a name. I chose ‘P2 digout’. Here is the completed circuit. The micro:bit’s P2 and GND pins are connected to the IN and GND pins of the relay board. This could be the basis of a home security system where the light gets switched on and off at certain times of the day when you go away on holiday. Remember this board will only switch voltages up to 12V DC, and isn’t suitable for connecting to the household AC mains!

Here is an example code for a Morse code tapper using the A button. So far we have only used digital output from the micro:bit to control a circuit. But we could simulate a dimmer switch by using button A to increase the brightness and button B to decrease it. We can use the same circuit, but this time use an analog write to pin P2 to send a value of between 0 and 1023.

We will need a variable to hold the current brightness level, say ‘strength’.
The `On start' block gives `strength' an initial value, and turns on the bulb. Every time button A is pressed it checks to see if `strength' is not yet too high to increase. If that’s OK then it increases `strength' by 20 and sends the new value to pin P2. Can you write the third block that makes button B dim the lamp provided `strength' won’t go below 0?

So far this has been all about control. We are using the micro:bit’s built-in buttons to control whether a lamp is on or off, like a switch, and also to vary its strength, like a dimmer (or potentiometer). Now let’s see how a sensor can be used to control an output. We have just seen that the MakeCode editor allows us to write numbers between 0 and 1023 to the micro:bit’s pins to provide analog outputs. Of course it works the other way as well. The next little program just picks a random whole number between 0 and 1023 and uses it to output a corresponding number of LEDs on the simulator.

`pick random' comes from the `Math' menu, and `plot bar graph' comes from the `Led' menu. Each time you press button A you should see a pattern of between 0 and 25 LEDs show on the display. Press button B after each to clear the screen. A new feature of the MakeCode editor has appeared below the simulator’s display of the micro:bit. Let’s explore how the oval bar labelled `Show data' works. The next little program just uses one of the micro:bit’s on-board sensors, the accelerometer.
If you run this in the simulator you can simulate shaking the micro:bit for 10 seconds and generating 100 patterns on the LED display. Now try clicking on the ‘Show data’ bar, press button A, and start ‘shaking’ the micro:bit image. On the right of the screen where your program was, there is now a moving graph of the data being streamed from the accelerometer. In this case we can see that it ranges between -806 and 1023 – which is because acceleration can be positive or negative depending on the direction you are moving the micro:bit in. The explanation of how to use the ‘Viewing data’ feature is [here].

The ‘Basic’ ‘show number’ block can only show one of the number’s digits at a time, and so can be hard to read. There is a useful tool to ‘map’ a sensed value, like the x-accelerometer reading between -1023 and +1023, to the range of single digit numbers 0, 1, …, 9. The tool is found in the ‘Advanced’ ‘pins’ menu.

Later we will see how we can use MakeCode’s ‘Viewing data’ feature to send data from sensors attached to a micro:bit through the USB cable for display and saving on a computer. First we will do some more explorations about how to use sensors attached to the micro:bit’s pins. So now let’s bring the sensor board into play so that the strength of the light can be controlled by one of the external sensors for light, sound or temperature.
You need to connect the GND and 3V pins on the micro:bit to either of the corresponding inputs on the Sensor Board. Also the Pin P1 to the right-hand input above the light sensor.

When you run the test program above on the simulator, you can use the mouse on pin P1 to drag the ‘light level’ up and down. It will show a value in white between 0 and 1023. When you press button A the corresponding single digit value between 0 and 9 will be displayed.

Can you now combine the ideas from this section to make the light bulb shine more brightly when the sensed light value is low, and vice versa?

We’ll finish this session by powering up an electric motor. I’m using a 6V motor as supplied by Kitronik for their line-following buggy, but you could use Lego, Meccano or K’Nex motors as well.
The Relay board is connected to the GND and P0 pins of the micro:bit. The motor is connected to the battery pack and the OUT pins of the Relay board. The simple program just drives the motor at half power when the A button is pressed and switches it off when the B button is pressed. Can you now put a variable resistor (potentiometer) into the system to control the motor’s speed? Maybe you could simulate a driverless car by including a proximity sensor in the circuit which would make the motor cut out?

If you want to make a robot buggy you will need to control a pair of motors. A very helpful device is a Motor Driver Board, such as that sold by Kitronik for £12.78. The 6V external battery pack is connected by red and black leads to the pair of power inputs on the left side of the board. The micro:bit is plugged into a slot on the board called an `edge connector’ which links the I/O pins of the micro:bit to the board. This also provides a 3V power source for the micro:bit. There are four pairs of connectors on the front of the board. The two left-hand blocks each control a motor. I have connected the left-most block to a Kitronik 6V motor and wheel with brown/green and white/yellow leads. A motor, wheel and tyre costs £3.72. Motor1 is controlled through pins P8 and P12. The example program will run the motor one way when A is pressed, and the other way when B is pressed. When both A and B are pressed the motor stops.

As well as sending digital outputs to the board through pins 8 and 12, we can also write analog values as well to control the speed of the motor. The change above makes the motor turn slowly when A is pressed.
You can also install an additional package from Kitronik to add a couple of useful blocks to control the motors.

Use this url to add the blocks: https://github.com/KitronikLtd/pxt-kitronik-motor-driver

In the next sections we will build up more sophisticated ways of controlling systems by getting the micro:bit to talk to other devices, like mobile phones, game controllers, other micro:bits and PCs. You’ll find an example video here. See what other information you can find out online about how people are using micro:bits to control systems. Here is information about one remote controlled robotic vehicle.