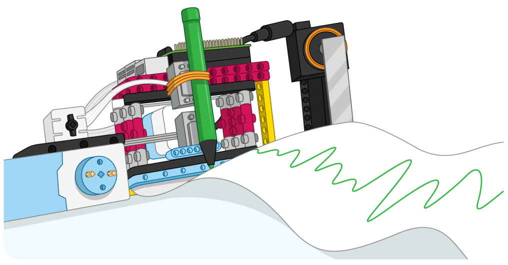
Projects

LEGO® data plotter

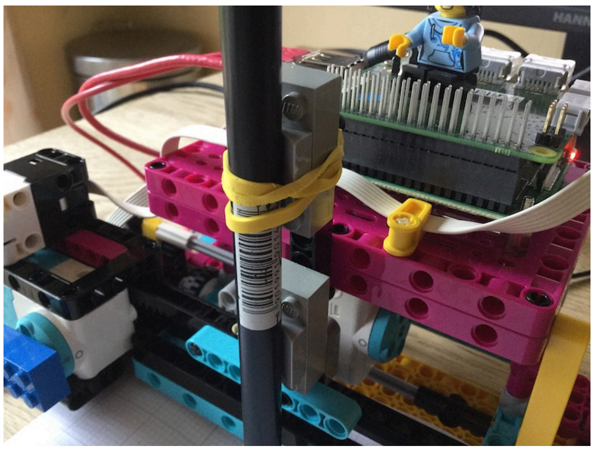
Plot data with pen and paper, using LEGO® and the Build HAT



Step 1 Introduction

Use LEGO® and the Raspberry Pi Build HAT to create a data plotter.

What you will make



What you will learn

- How to calculate angles of rotation
- How to map data ranges onto appropriate scales for visualisation
- How to use conditional statements (if/else)

Hardware

- A Raspberry Pi computer
- A Raspberry Pi Build HAT
- Two LEGO® Technic™ motors
- A LEGO® SPIKE™ Force Sensor OR a push button, breadboard, and jumper wires
- Assortment of LEGO®, including two small wheels (we used a selection from the LEGO® Education SPIKE™ Prime kit (https://education.lego.com/en-gb/product/spike-prime))
- A 7.5V power supply with a barrel jack (you could instead use a battery pack, but make sure that all cells are fully charged)

Software

- Python 3
- The Vcgencmd Python3 library

Downloads

- LEGO® SPIKE™ Prime building instructions: *Track Your Parcels* (1/2) (https://le-www-live-s.legocdn.com/sc/media/lessons/prime/pdf/building-instructions/track-your-packages-bi-pdf-book1of2-05883f81fed73ac3738781d084e0d4e2.pdf)
- LEGO® SPIKE™ Prime building instructions: *Track Your Parcels* (2/2) (https://le-www-live-s.legocdn.com/sc/media/lessons/prime/pdf/building-instructions/track-your-packages-bi-pdf-book2of2-80dc3c8c61ec2d2ffa785b688326ef74.pdf)
- Finished script for Lego Plotter (http://rpf.io/p/en/lego-plotter-go)

Install the Vcgencmd python library

Make sure you are connected to the internet.

Open the terminal on your Raspberry Pi by pressing `Ctrl+Alt+T` on your keyboard.

At the prompt type: `pip3 install vcgencmd` and press `Enter`.

Wait for the confirmation message (it won’t take long) then close the terminal window.

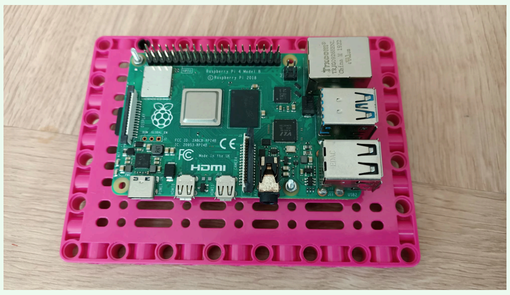
Additional information for educators

You can download the completed project here (http://rpf.io/p/en/projectName-get).

If you need to print this project, please use the printer-friendly version (https://projects.raspberrypi.org/en/projects/projectName/print).

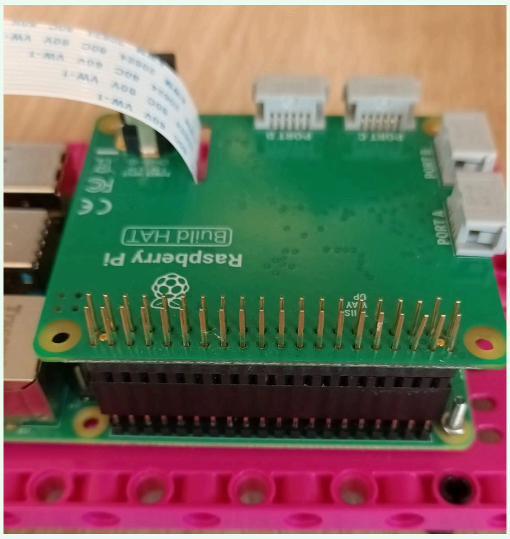
Before you begin, you’ll need to have set up your Raspberry Pi computer and attached your Build HAT:

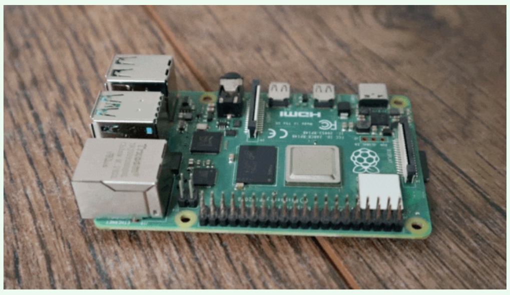
Mount your Raspberry Pi on to the LEGO Build Plate using M2 bolts and nuts, making sure the Raspberry Pi is on the side without the ‘edge’:



Mounting the Raspberry Pi this way round enables easy access to the ports as well as the SD card slot. The Build Plate will allow you to connect the Raspberry Pi to the main structure of your dashboard more easily.

Line up the Build HAT with the Raspberry Pi, ensuring you can see the `This way up` label. Make sure all the GPIO pins are covered by the HAT, and press down firmly. (The example uses a stacking header (https://www.adafruit.com/product/2223), which makes the pins longer.)





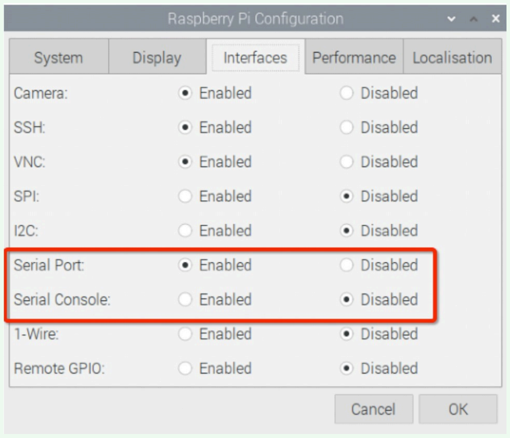
You should now power your Raspberry Pi using the 7.5V barrel jack on the Build HAT, which will allow you to use the motors.

If you have not already done so, set up your Raspberry Pi by following these instructions:

Setting up your Raspberry Pi (https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up)

Once the Raspberry Pi has booted, open the Raspberry Pi Configuration tool by clicking on the Raspberry Menu button and then selecting “Preferences” and then “Raspberry Pi Configuration”.

Click on the “interfaces” tab and adjust the Serial settings as shown below:



Camera:	[✓ Enabled]	Disabled
SSH:	[✓ Enabled]	Disabled
VNC:	[✓ Enabled]	Disabled
SPI:	Enabled	[✓ Disabled]
I2C:	Enabled	[✓ Disabled]
Serial Port:	[✓ Enabled]	Disabled
Serial Console:	Enabled	[✓ Disabled]
1-Wire:	Enabled	[✓ Disabled]
Remote GPIO:	Enabled	[✓ Disabled]

You will also need to install the buildhat python library by following these instructions:

Install the buildhat Python library

Open a terminal window on your Raspberry Pi by pressing `Ctrl+Alt+T`.

At the prompt type: `sudo pip3 install buildhat`

Press `Enter` and wait for the “installation completed” message.

Step 2 Move the motors with data

You may have seen in earthquake disaster movies a scene where a seismometer (https://en.wikipedia.org/wiki/Seismometer) is used to show the tremors. The design of such devices is quite simple: one motor is used to move the paper past the pen (the x-axis), while another, at a right-angle to the first, moves the pen in response to the changing data (y-axis).

In this project, you will create a plotter from LEGO®, and connect it to your Raspberry Pi so it can plot real-time data.

Connect a monitor, keyboard, and mouse to your Raspberry Pi. If you've never used a Raspberry Pi before, you might want to start with this project (https://projects.raspberrypi.org/en/projects/raspberry-pi-getting-started).

Attach the Build HAT to your Raspberry Pi (make sure you can see the Raspberry Pi logo on the top) and connect a 7.5V power supply to the barrel jack of the Build HAT. This will boot your Raspberry Pi.

Open Thonny from the programming menu, and add the following lines to begin your program by importing the libraries you will be using:

plotter.py

```
from random import randint
from time import sleep
from buildhat import Motor
```

Save this program as `plotter.py` by pressing `Ctrl+s`.

Now use the `randint` function to create a random value between a range (in this case, -180 to 180) and store it in a variable called `new_angle`:

plotter.py

```
new_angle = randint(-180,180)
print(new_angle)
```

Run your program a few times by clicking the **Run** button at the top of the window. You should see different values appear in the shell beneath your code each time.

Instead of running this script manually, create a **loop** to run the script repeatedly. To run the same lines continuously, you can use a `while True:` loop.

Add a blank line above the code you just added by pressing `Enter`.

On this new line, enter `while True:`; make sure you have a capital ‘T’.

plotter.py

```python
while True:
new_angle = randint(-180,180)
print(new_angle)
```

Add four spaces to the start of each of the lines beneath to create an **indented code block**. This tells the computer which lines are included in your loop.

plotter.py

```python
while True:
    new_angle = randint(-180,180)
    print(new_angle)
```

At the end of your code, press `Enter` to add another indented line. On this line, type `sleep(0.1)`.

plotter.py

```python
while True:
    new_angle = randint(-180,180)
    print(new_angle)
    sleep(0.1)
```

Run your code to see the values printed in the shell. If you get any errors, check that your code looks like this:

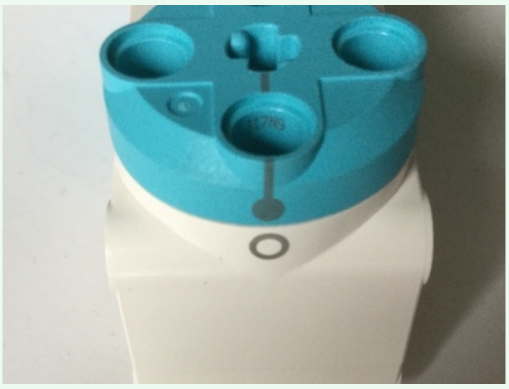
plotter.py

```python
from random import randint
from time import sleep
from buildhat import Motor

while True:
    new_angle = randint(-180,180)
    print(new_angle)
    sleep(0.1)
```

Now that you have some data, you can use this to control the position of a motor.

Connect a LEGO® Technic™ motor to port A on the Build HAT. Add some additional LEGO elements to the motor axle so that it is easy to see the motor turning.

Line up the element with the line mark on the motor and then set the motor to the zero position:



Now, modify the main body of your program so that the angle turned to by the motor is the same as the latest value produced by your simulated sensor.

To do this, you need to set up your motor so it can be accessed by the program.

Create a `motor_y` object for port `A` on the Build HAT and then turn the motor to the `0` position with a speed of `100`.

plotter.py

```python
motor_y = Motor('A')
motor_y.run_to_position(0, 100)
```

The next line makes the motor turn to the angle stored in `new_angle`.

plotter.py

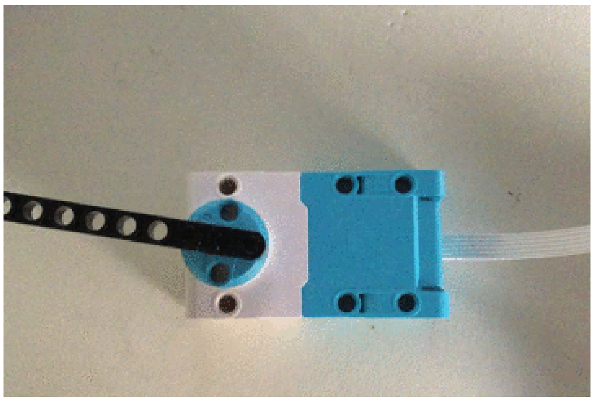
```python
while True:
    new_angle = randint(-180,180)
    print(new_angle)
    motor_y.run_to_position(new_angle, 100)
```

Click **Run** and you should see your motor spin clockwise to different positions in response to the changing data. If you run the program again, it should reset the motor position back to `0` before moving randomly again.

If you get errors, then check your code looks like this.

plotter.py

```python
from random import randint
from time import sleep
from buildhat import Motor

motor_y = Motor('A')
motor_y.run_to_position(0, 100)

while True:
    new_angle = randint(-180,180)
    print(new_angle)
    motor_y.run_to_position(new_angle, 100)
    sleep(0.1)
```



Save your project

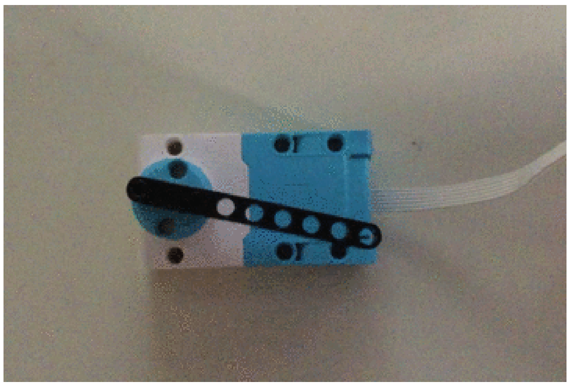
Step 3 Create a plot range

In this step we will control the direction in which the motors move (clockwise or anti-clockwise) to set a maximum point of travel in each direction.

Why you need to change the way the motors move

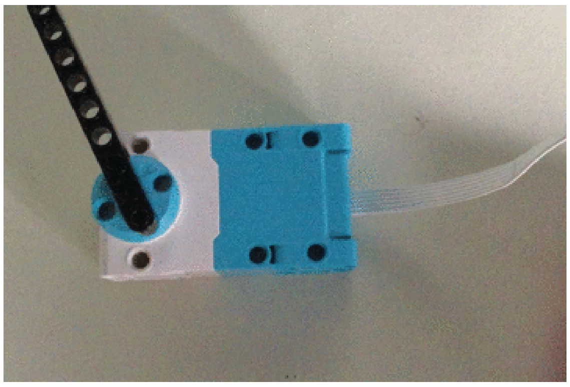
Your motor will always take the shortest path to the new position.

For example, if the motor is at 170 degrees and the next position is -170 degrees, it will travel in a clockwise direction, passing through the 180 degree position in order to get to its destination as quickly as possible.



This is fine for our simulation, but our plotter will not have this freedom of movement. Once the pen has reached the top or bottom of the paper (y-axis), it cannot continue to travel up to emerge at the bottom — it will break. So your plotter will need to be prevented from travelling clockwise past the 180 degree mark.

This can be achieved by altering the behaviour of the motor when moving to a position. You can do this by passing an additional `direction=` parameter to the `run_to_position()` function. You can set this value to `"clockwise"`, `"anticlockwise"`, or `"shortest"`, which is the default 'shortest path' behaviour.



So, for example, `motor_y.run_to_position(50, 100, direction="anticlockwise")` will drive a motor to the 50 degrees position, turning anti-clockwise at maximum speed.

It is possible to add a **conditional check** to your loop to ensure that the motor never passes through 180 degrees and always moves from a higher angle to a lower one by turning in an anti-clockwise direction.

You can find the last position of the motor by using `motor_y.get_aposition`.

Check for the motor’s current angle at the top of your `while` loop.

plotter.py

```python
while True:
    current_angle = motor_y.get_aposition()
    new_angle = randint(-180, 180)
    print(new_angle)
    motor_y.run_to_position(new_angle, 100)
    sleep(0.1)
```

Now, in the `while` loop, you can add a check to see if the current value of `new_angle` is greater or less than the `current_angle`.

plotter.py

```python
while True:
    current_angle = motor_y.get_aposition()
    new_angle = randint(-180, 180)
    print(new_angle)
    if new_angle > current_angle:
        motor_y.run_to_position(new_angle, 100,
direction="clockwise")
        print('Turning CW')
    elif new_angle < current_angle:
        motor_y.run_to_position(new_angle, 100,
direction="anticlockwise")
        print('Turning ACW')
    sleep(0.1)
```

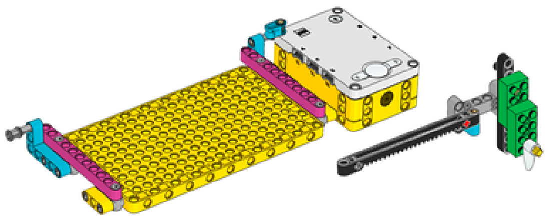
Run your code. These conditional tests will prevent the motor from changing from a negative value to a positive one by passing through 180 degrees (and vice versa).

Save your project

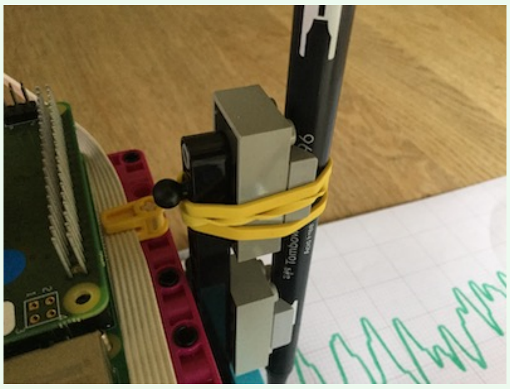
Step 4 Build the plotter

In this step, you will build an x/y plotter using LEGO®.

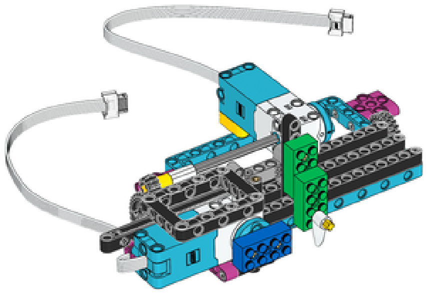
There are plenty of ways you could do this, but the build instructions for the LEGO® SPIKE™ Prime *Track Your Parcels* project are a great starting point. You can use the motor from the previous step for the y-axis motor (the one holding the pen) in the build.



You will need to adapt the build slightly so that the arm is able to hold a pen. Rubber bands are a great way to hold a pen snugly against LEGO.



The second part of the build completes the mechanism that uses the two motors to drive the plotter.



Connect the LEGO® Technic™ motor that drives the pen up and down to port A on the Build HAT.

Now you can use your simulated data source to test your plotter. For now, keep the lid on your pen or remove it all together while you observe the motion caused by the data.

Calibrate the plotter

Your program currently allows the motor to move through its full range of motion (-180 to +180 degrees from the zero point). But the physical constraints of the plotter mean that if you tried to drive the toothed rail to its maximum and minimum positions, it would crash the pen arm into other parts of the build. In order to avoid this, you must centre the bar.

Click into the **Shell pane** of Thonny (the window beneath the code) so that you can execute Python one line at a time.

Enter these lines into the **Shell** (you can just copy and paste them from your program above) pressing `Enter` between each one:

```
>>> from buildhat import Motor
```

Press `Enter`.

Type:

```
>>> motor_y = Motor('A')
```

Press `Enter`.

Type:

```
>>> motor_y.run_to_position(0, 100)
```

Press `Enter`.

This should centre or **zero** your motor.

Adjust the position of your pen arm by gently pushing the toothed bar to the middle of its path, so that the pencil or pen lines up with the other motor.

Save your project

Step 5 Feed in paper

You will now program the second motor to feed paper through the plotter at a constant rate.

Feed a sheet of A5 paper (or cut up some scrap to about this size) underneath the small wheels from behind.

Plug the rear LEGO® Technic™ motor (which drives these wheels) into port B on the Build HAT.

Create an object called `motor_x` for this motor, below the similar line for `motor_y`:

plotter.py

```
motor_y = Motor('A')
motor_x = Motor('B')
motor_y.run_to_position(0, 100)
```

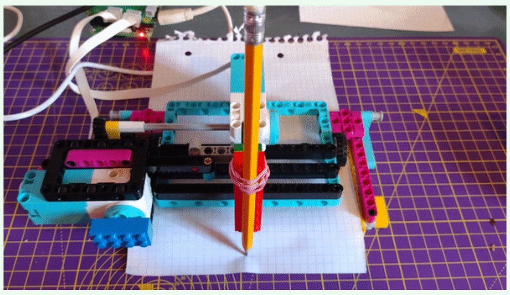
Add a line to start this motor turning immediately before the `while True` loop:

plotter.py

```
motor_y = Motor('A')
motor_x = Motor('B')
motor_y.run_to_position(0, 100)
motor_x.start(-25)
```

This will make the feeder motor run at a constant rate of -25 turns per minute when the program starts. Change the number in the brackets to experiment with the speed.

Run your code and watch the paper being fed through the plotter, as the pencil moves randomly in the `y` direction.



To stop the motor feeding the paper, you can type the following into the **Shell**.

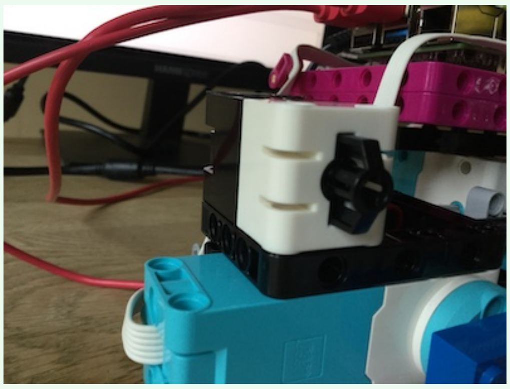
```
>>> from buildhat import Motor
>>> motor_x = Motor('B')
>>> motor_x.stop()
```

Save your project

Step 6 Add a button control

To stop and start the plotter running, you can add a button to your build.

The LEGO® SPIKE™ Prime Force Sensor can act as a simple button. Connect one to port C on your Build HAT.



Edit your `plotter.py` program to include a button control. Add a comma followed by `ForceSensor` (making sure you include **both** capital letters!) to the end of the line that says `from buildhat import Motor`:

plotter.py

```python
from random import randint
from time import sleep
from buildhat import Motor, ForceSensor
```

Add this line to create an object for the button after the similar lines for the motors:

plotter.py

```
motor_y = Motor('A')
motor_x = Motor('B')
button = ForceSensor('C')
motor_y.run_to_position(0, 100)
motor_x.start(-25)
```

Change your main loop from `while True` to:

plotter.py

```
while not button.is_pressed():
    current_angle = motor_y.get_aposition()
    new_angle = randint(-180, 180)
```

Now you can stop the plotter operating by pressing the button. To tidy everything up and stop both motors, add the following lines at the end of your program.

plotter.py

```
    elif new_angle < current_angle:
        motor_y.run_to_position(new_angle, 100, direction="anticlockwise")
        print('Turning ACW')
    sleep(0.1)

motor_x.stop()
motor_y.run_to_position(0, 100)
```

Now you are ready to test your plotter. Your final script should look like this:

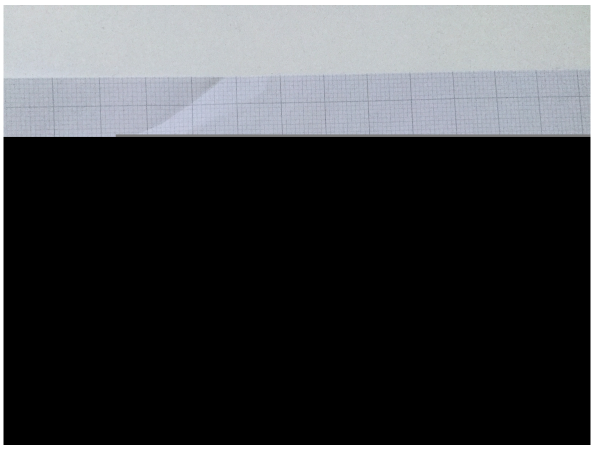
plotter.py

```python
#!/usr/bin/python3
from random import randint
from time import sleep
from buildhat import Motor, ForceSensor

button = ForceSensor('C')
motor_y = Motor('A')
motor_x = Motor('B')

motor_y.run_to_position(0, 100)
motor_x.start(speed=-25)

while not button.is_pressed():
    current_angle = motor_y.get_aposition()
    new_angle = randint(-180, 180)
    if new_angle > current_angle:
        motor_y.run_to_position(new_angle, 100, direction="clockwise")
        print('Turning CW')
    elif new_angle < current_angle:
        motor_y.run_to_position(new_angle, 100, direction="anticlockwise")
        print('Turning ACW')
    sleep(0.1)

motor_x.stop()
motor_y.run_to_position(0, 100)
```

Feed a piece of paper from the back of the plotter so that the front short edge is just beyond the pen.

Start the program in Thonny, and watch as the pen plots your random data on your paper!

Once the paper has been used, press the Force Sensor button to stop everything.



In the next step, you will use a real-time data source for your input data!

Save your project

Step 7 Add a real-time data source

There are are a huge variety of sensors you could add to your Raspberry Pi to provide a data feed for your plotter.

Let’s start with an in-built data source: the temperature of the CPU on the Raspberry Pi itself. If you haven’t installed the `vcgencmd` library, you should do that now.

Install the Vcgencmd python library

Make sure you are connected to the internet.

Open the terminal on your Raspberry Pi by pressing `Ctrl+Alt+T` on your keyboard.

At the prompt type: `sudo pip3 install vcgencmd` and press `Enter`.

Wait for the confirmation message (it won’t take long), then close the terminal window.

Using the **Shell/REPL** in Thonny, enter the following lines of Python to test and read the CPU temperature.

```
>>> from vcgencmd import Vcgencmd
```

Press `Enter`.

Type:

```
>>> vcgm = Vcgencmd()
```

Press `Enter`.

Type:

```
>>> vcgm.measure_temp()
```

Press `Enter`.

You should see the **Shell** return a number value (it should be somewhere around 50) — this is how hot your CPU is running.

Now let's warm things up by getting the CPU to do some work!

Open the web browser and watch a YouTube video. After a few seconds, go back to Thonny and re-run the last line of Python and you should see that the temperature has increased.

Now that you've seen how to read the temperature of the CPU with Python, you can modify your `plotter.py` program so that it uses this as its data source.

First, underneath the existing import lines at the top of the file, add the lines to import the Vcgencmd library:

plotter.py

```python
from random import randint
from time import sleep
from buildhat import Motor, ForceSensor
from vcgencmd import Vcgencmd
```

Create a vcgencmd object:

plotter.py

```python
from random import randint
from time import sleep
from buildhat import Motor, ForceSensor
from vcgencmd import Vcgencmd

motor_y = Motor('A')
motor_x = Motor('B')
button = ForceSensor('C')
vcgm = Vcgencmd()

motor_y.run_to_position(0, 100)
motor_x.start(-25)
```

Change the program so that it uses real-time temperature values rather than randomly generated numbers. To do this, you need to replace `randint(-180, 180)` with `vcgm.measure_temp()`.

plotter.py

```python
while not button.is_pressed():
    temp = vcgm.measure_temp()
    current_angle = motor_y.get_aposition()
```

Before you can use the temperature of the Raspberry Pi’s CPU as a data source for your plotter, you want to make sure that the maximum possible value produced by the data source will be mathematically converted so that it fits on a scale between -180 and 180.

The range of temperature values produced by `vcgencmd` should be from around 50°C (when the Raspberry Pi is on, but not doing very much) to less than 90°C when working hard (at 85°C, the Raspberry Pi will throttle its performance to keep the temperature below this value). Let’s say you want to plot a range from 40°C to 90°C — you need to map this to your available values: -180 to 180.

You can create a function to remap one range of values to another range of values.

Add this function above your `while` loop. It will take a temperature range and an angle range, and then remap the temperature into an angle.

plotter.py

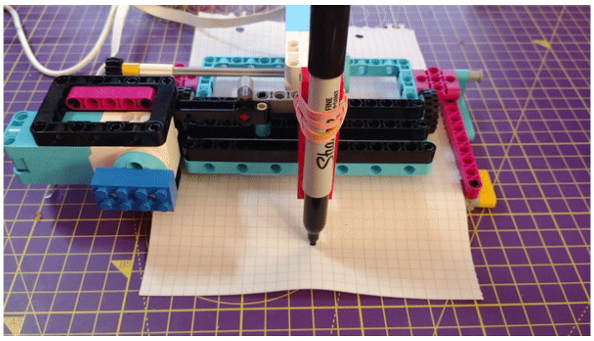
```python
def remap(min_temp, max_temp, min_angle, max_angle, temp):
    temp_range = (max_temp - min_temp)
    motor_range = (max_angle - min_angle)
    mapped = (((temp - min_temp) * motor_range) / temp_range) +
min_angle
    return int(mapped)
```

Now, in the `while` loop, you can use this function to calculate a new angle for the motor to turn to.

plotter.py

```python
while not button.is_pressed():
    temp = vcgm.measure_temp()
    current_angle = motor_y.get_aposition()
    new_angle = remap(50, 90, -170, 170, temp)
```

Now you can run your program. Make the Raspberry Pi CPU get warmer like you did before and you should see the pen gradually move upwards. Feel free to change the `min_temp` and `max_temp` parameters, if your pen isn't moving too much.

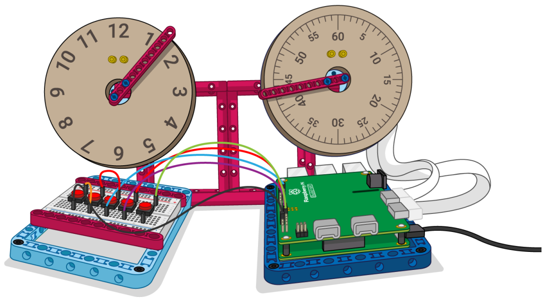


Save your project

Step 8 What next?

- Add a LEGO® Technic™ Color Sensor to detect when the paper has run out.
- Modify the LEGO® Technic™ Force Sensor code so that the speed of the motor controlling the paper feed is increased by increasing the pressure on the sensor.
- Add other data sources. You could use another external sensor or a feed for an online data stream.

If you are following the Introduction to LEGO BuildHAT (https://projects.raspberrypi.org/en/pathways/lego-intro) pathway, you can move on to the LEGO Data Dash (https://projects.raspberrypi.org/en/projects/lego-data-dash) project. In this project, you will make some dials and readouts with LEGO and get them to display the air quality data from anywhere in the world!



If you want to have more fun exploring python, then you could try out any of these projects (https://projects.raspberrypi.org/en/projects?software%5B%5D=python).