IO1

Minecraft Pi & Physical Computing Blocks

Deliverable: IO1A2

STEM4CLIM8

29 September, 2021

HeartHands Solutions LTD

Authored by: HeartHands Solutions LTD

Project Number: 2020-1-UK01-KA201-079141

Co-funded by the Erasmus+ Programme of the European Union

Executive Summary

In a recent OECD survey (OECD- Education and Skills Today 2018), covering 25 European countries, almost all countries report shortfalls of skills that teachers need to meet school needs, combined with difficulties in updating teachers' skills. However, digitisation is expected to profoundly change the way we learn and work. Many children entering school today are likely to end up working in jobs that do not yet exist. Preparing students for these uncharted territories means that we not only have to make sure that they have the right technical capabilities but that we must strengthen their emotional and social skills. Resilience, the individual capacity to overcome adverse circumstances and use them as sources for personal development, lies at the core of being able to successfully adapt to change and thus actively engage with our digital world. At the same time, we need to acknowledge Internet addiction and behaviours at risk of IAB (Internet Addiction Behaviour) as emerging problems for our youth. A STEM approach bridging physical computing with environmental consciousness while focusing on off-screen collaborative activities is an excellent way of improving technical capabilities while strengthening emotional and social skills.

STEM4CLIM8 has as primary objective to produce approaches and tools to help those working with children reach out to them with a view to help them engage with programming and develop STEM related skills. It aims to achieve this not by increasing screen time but by encouraging hands on play through the creation of a custom virtual world using Minecraft modding and the execution of missions dealing with natural disasters and using physical computing blocks which will be programmed to interact with the virtual world through the Raspberry GPIO. The missions will reveal the science behind natural phenomena frequently associated to climate change and inspire environmental consciousness while at the same time enhance STEM skills.

Reference:

*OECD- Education and Skills Today, Succeeding with resilience-Lessons for schools, January 29, 2018, Retrieved February 18, 2021 from: https://oecdedutoday.com/succeeding-with-resilience-lessons-for-schools/

Table of Contents

1. Physical Computing Blocks in Minecraft Pi

A set of DIY physical computing blocks have been designed to interact with a Minecraft Pi World where a series of natural disasters occur. These blocks will be connected through the Raspberry Pi GPIO and will be programmed through Python programming language in order to interact with the Minecraft Pi world, simulating natural disaster and their catastrophic results to a small "Fishing Town".

This guide, addressed to educators, offers instructions for assembling the physical computing blocks, for connecting them to the GPIO and for programming them using Python programming and Minecraft Pi.

Teachers should be equipped with the following:

- STEM4CLIM8 console
- Physical Computing Blocks
- Educational Material for teaching these phenomena.

1.1 Installing "Fishing Town" Minecraft Pi world

In order to execute any of the natural disasters' scenarios, you first need to install a map of a fishing town that will be used as the testing ground of the different disasters. To do so, you first need to turn on your Raspberry Pi on your STEM4CLIM8 console.

On the desktop, you will see a folder named "Fishing Town Map". Double click on it to see the contents of the folder. Then follow these simple steps:

1. Open Minecraft Pi and create a new world, then close Minecraft Pi
2. Open "File Manager".
3. Click on "View" and tick "Show Hidden".
4. Double click on the folder named ".minecraft".
5. Double click on the folder named "games".
6. Double click on the folder named "com.mojang".
7. Double click on the folder named "minecraftWorlds".
8. Copy the contents of the "Fishing Town Map" folder into an existing "world" folder.
9. Rename that folder to "fishingtown".
10. Open Minecraft Pi and load the "fishingtown" world.

A "Fishing Town" Minecraft Pi world will be loaded. Please note, that you need to redo this procedure every time you need a fresh and disaster-free map.

This Minecraft Pi world will be used to simulate the following natural disasters:

- Earthquake
- Volcano eruption
- Sinkhole
- Heatwave
- Flood

Co-funded by the Erasmus+ Programme of the European Union

1.2 Scenarios and Interactions

With the use of the Raspberry Pi GPIO, sensors, electronics, peripherals and DIY carton constructs, students will learn about physical disasters, test their results and discuss their consequences.

At first, teachers will guide their students on how to connect the sensors and other electronics to the Raspberry Pi, how to build the DIY carton constructs and how to run the Minecraft Pi activities in Python.

The Minecraft Pi world is formatted as a small fishing town in which the physical disasters will occur. The activities are categorised into five topics:

1. Earthquake
2. Sinkhole
3. Volcano eruption
4. Heatwave
5. Flood

For each activity, specific instructions on how to execute it have been developed. It is recommended to split your students in several groups, and each group should have access to at least one STEM4CLIM8 console, a connected keyboard and mouse as well a speaker.

1.3 Earthquake

1.3.1 Equipment and scenario

This activity will present an earthquake to the Minecraft Pi world. Students will connect a button to the Raspberry Pi GPIO and program it to create an earthquake after a press.

The idea is to create a physical computing block using the following electronics and peripherals:

1. 1 x Breadboard
2. 1 x Push button with button cap
3. 1 x 220 Ohm Resistor
4. 2 x Male-to-Female Jumper cables
5. 1 x Carton construct

After everything is connected, the students should load the "Fishing Town" map and the Python script named "earthquake.py" which can be found in the desktop of each STEM4CLIM8 console.

Then each time the button is pressed, an earthquake will happen in the Minecraft Pi world which will destroy buildings and other blocks of the "Fishing Town", similar to a real earthquake of high magnitude.

A follow-up discussion with your students on the impact of the earthquake and the meaning of well-constructed building should follow after the activity.

1.3.2 Electronic circuit and physical block

Connecting a push button to the Raspberry Pi's GPIO is a very simple process as depicted in Figure 1 below:

1. Connect the right side of the push button to input pin 16 (GPIO23) using a male-to-female jumper cable.
2. Connect the left side of the push button to the 220 Ohm resistor.
3. Connect the other side of the resistor to 3.3V pin 1.
4. You circuit is ready.

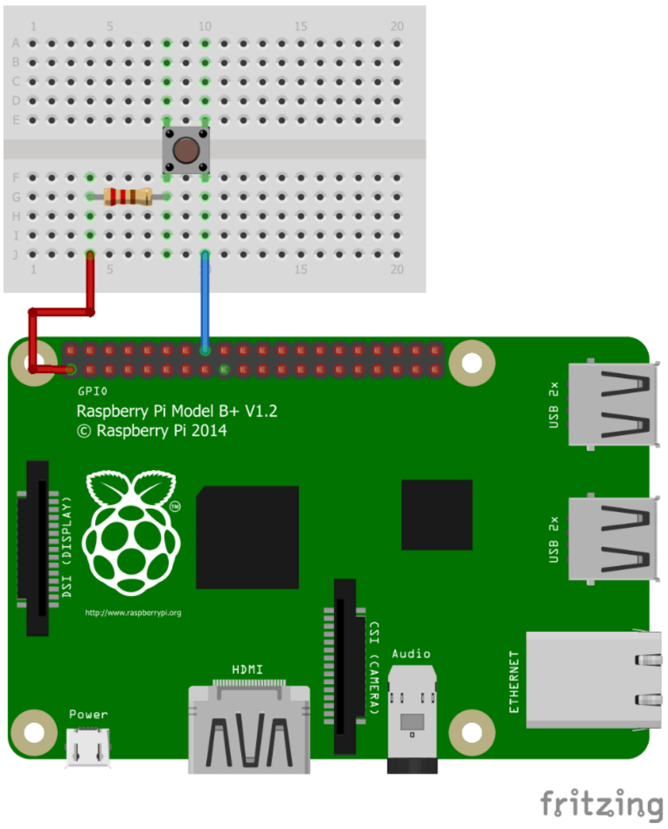

Figure 1 Connecting a push-button to the GPIO using a breadboard and two jumper cables.

Understanding the different GPIO pins, you should take a look at Figure 2 which describes each pin's properties.

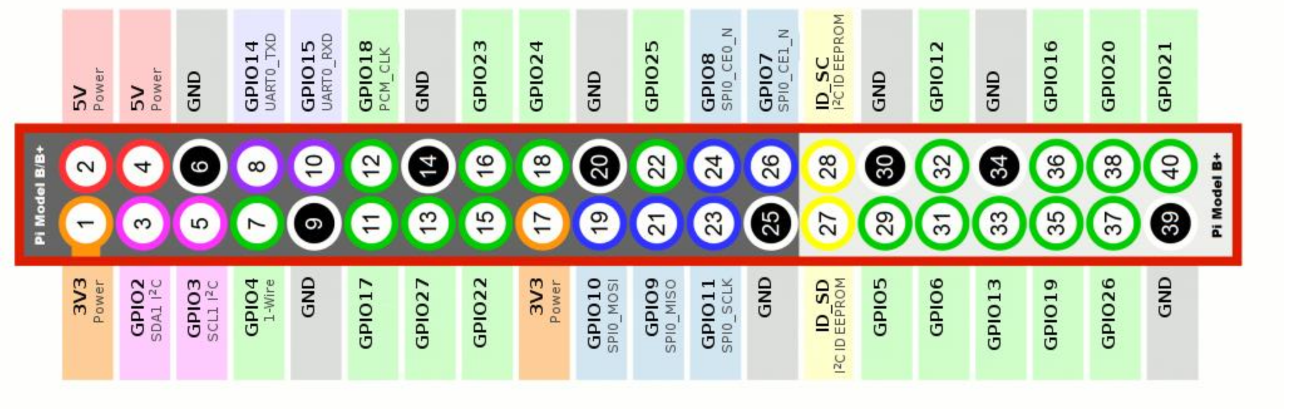

Figure 2 Raspberry Pi GPIO Pins

After creating the circuit, you need to assembly a DIY carton construct and place the circuit in it. For that purpose, you need to use the following printable image. For a better experience, make sure to print the construct in A4 white carton paper and then cut the specified edges.

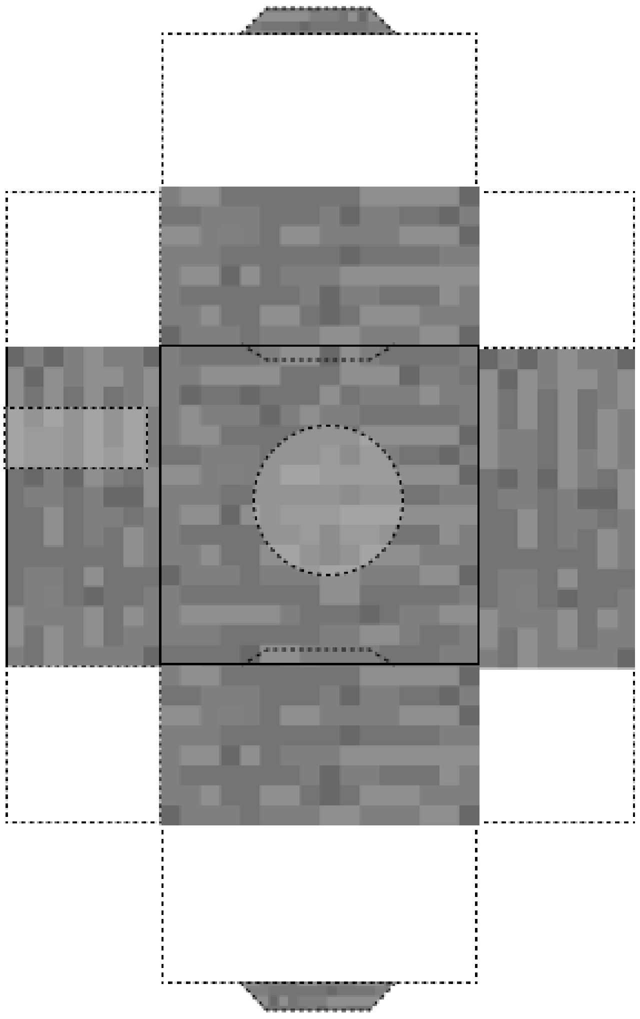
Figure 3 Printable carton construct for physical computing block with button.

1.3.3 Python programming

Simply open Thonny Python on your Raspberry Pi desktop, then load the script called "earthquake.py" which can be found on the desktop or copy and paste the script directly to the Python compiler of your choice. If you decide to connect the push-button to a different GPIO pin, please make sure to change the GPIO number in the script below (lines 10 and 42).

```
import RPi.GPIO as GPIO
import mcpi.minecraft as minc
import mcpi.block as block
mc = minc.Minecraft.create()
import random, time

GPIO.setmode(GPIO.BCM)
GPIO.setup(23, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
```

```
def earthquake(x, z):
    mc.postToChat('Earthquake!')
    y = mc.getHeight(x, z)
    endtime = time.time() + 60
    nearthtime = time.time()
    while endtime > time.time():
        if time.time() > nearthtime:
            nearthtime = time.time() + 5
        ppos = mc.player.getPos()
        if ppos.x < x+100 and ppos.x > x-100:
            if ppos.y < y+100 and ppos.y > -60:
                if ppos.z < z+100 and ppos.z > z-100:
                    mc.player.setPos(ppos.x, ppos.y, ppos.z)
        bx = random.randint(x-100, x+100)
        by = y
        bz = random.randint(z-100, z+100)
        if mc.getHeight(bx, bz) > -50:
            by = mc.getHeight(bx, bz)
        if mc.getBlock(bx, by, bz) in [block.GLASS.id, block.GLASS_PANE.id]:
            mc.setBlock(bx, by, bz, block.AIR.id)
            continue
        mc.setBlock(bx, by, bz, block.GRAVEL.id)
        mc.setBlocks(bx, by-1, bz, bx, -60, bz, block.AIR.id)

disasters = [earthquake]
def main(disasters, mc):
    baseed = random.randint(1, 10000)
    mc.postToChat('Press button for earthquake!')
    while True:
        if GPIO.input(23) == GPIO.HIGH: #Look for button press
            t = random.randint(5, 60)
            t = 5
            time.sleep(t)
            random.seed(baseed + t)
            baseed = random.randint(1, 10000)
            random.shuffle(disasters)
            disaster = random.choice(disasters)
            ppos = mc.player.getTilePos()
            disaster(ppos.x, ppos.z)
try:
    import _thread as thread
except ImportError:
    import thread
thread.start_new_thread(main, (disasters, mc))
```

1.4 Sinkhole

1.4.1 Equipment and scenario

This activity will present a sinkhole to the Minecraft Pi world that will happen after an earthquake. Similar to the first activity, students will connect a button to the Raspberry Pi GPIO and program it to create an earthquake and a sinkhole after a press.

The activity uses the following electronics and peripherals:

1. 1 x Breadboard
2. 1 x Push button with button cap
3. 1 x 220 Ohm Resistor
4. 2 x Male-to-Female Jumper cables
5. 1 x Carton construct

After everything is connected, the students should load the “Fishing Town” map and the Python script named “sinkhole.py” which can be found in the desktop of each STEM4CLIM8 console.

Then each time the button is pressed, an earthquake will happen in the Minecraft Pi world, followed by a sinkhole which will destroy buildings and sink the ground like a real-life catastrophe.

A follow-up discussion with your students on the impact of earthquakes and sinkholes should follow after the activity.

1.4.2 Electronic circuit and physical block

Connecting a push button to the Raspberry Pi's GPIO is a very simple process as depicted in Figure 4 below:

1. Connect the right side of the push button to input pin 16 (GPIO23) using a male-to-female jumper cable.
2. Connect the left side of the push button to the 220 Ohm resistor.
3. Connect the other side of the resistor to 3.3V pin 1.
4. You circuit is ready.

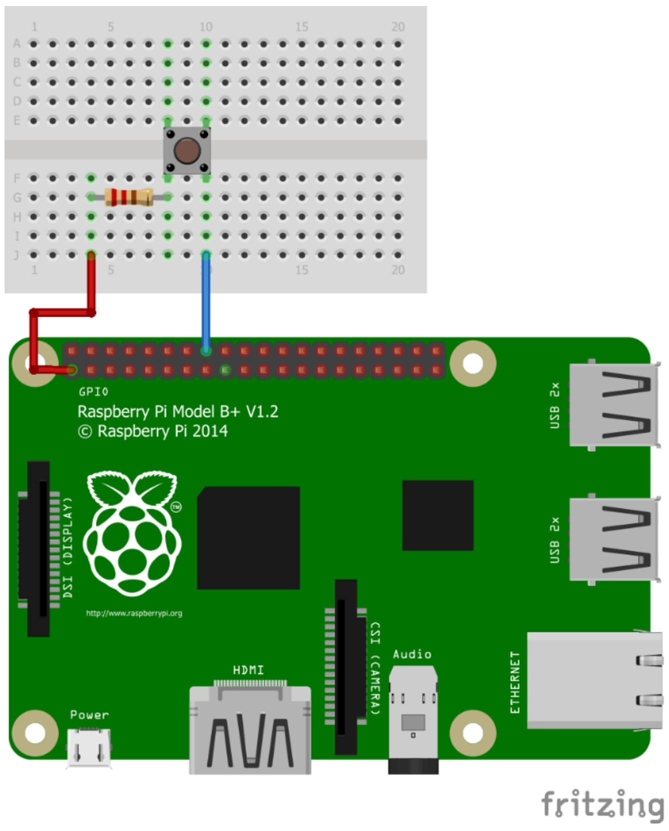

Figure 4 Connecting a push-button to the GPIO using a breadboard and two jumper cables.

Understanding the different GPIO pins, you should take a look at Figure 5 which describes each pin's properties.

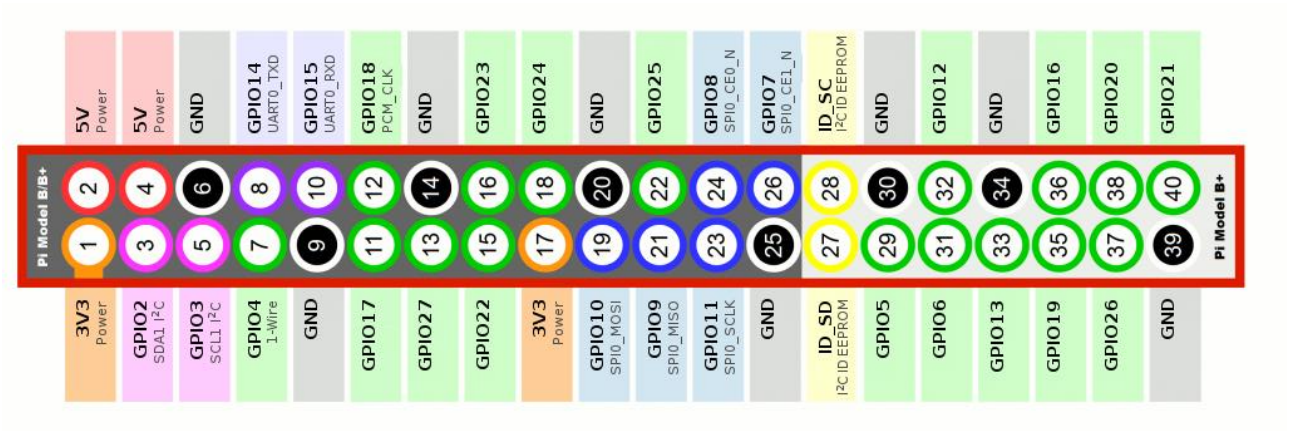

Figure 5 Raspberry Pi GPIO Pins

After creating the circuit, you need to assembly a DIY carton construct and place the circuit in it. For that purpose, you need to use the following printable image. For a better experience, make sure to print the construct in A4 white carton paper and then cut the specified edges.

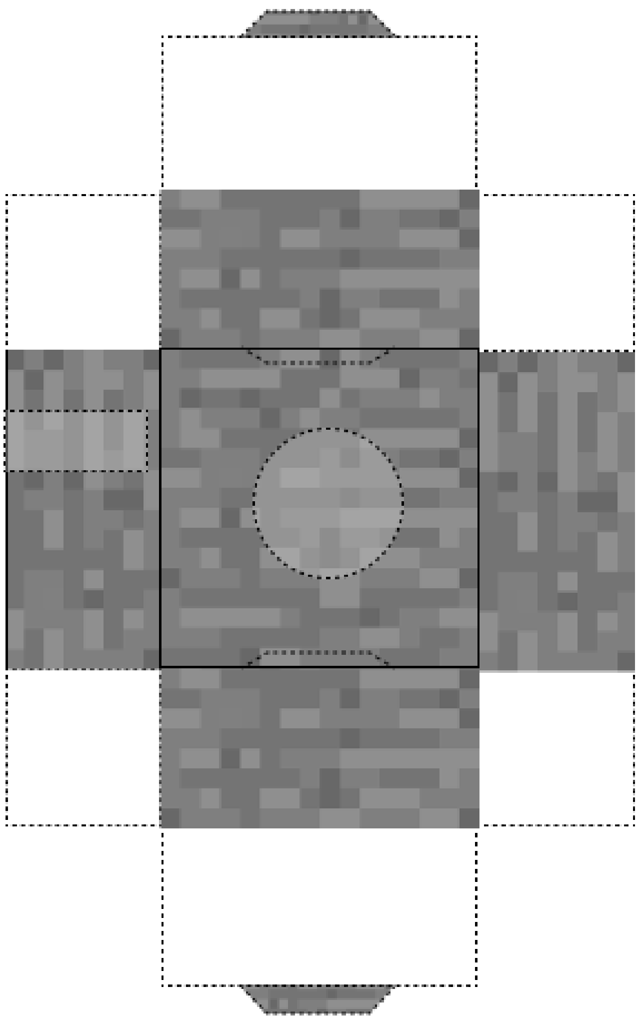

Figure 6 Printable carton construct for physical computing block with button.

1.4.3 Python programming

Simply open Thonny Python on your Raspberry Pi desktop, then load the script called "sinkhole.py" which can be found on the desktop or copy and paste the script directly to the Python compiler of your choice. If you decide to connect the push-button to a different GPIO pin, please make sure to change the GPIO number in the script below (lines 10 and 57).

```
import RPi.GPIO as GPIO
import mcpi.minecraft as minc
import mcpi.block as block
mc = minc.Minecraft.create()
import random, time

GPIO.setmode(GPIO.BCM)
GPIO.setup(23, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)

def sinkhole(x, z):
    mc.postToChat('Earthquake!')
    y = mc.getHeight(x, z)
    endtime = time.time() + 60
    nearthtime = time.time()
    while endtime > time.time():
        if time.time() > nearthtime:
            nearthtime = time.time() + 5
        ppos = mc.player.getPos()
        if ppos.x < x+50 and ppos.x > x-50:
            if ppos.y < y+50 and ppos.y > -60:
                if ppos.z < z+50 and ppos.z > z-50:
                    mc.player.setPos(ppos.x, ppos.y, ppos.z)
        bx = random.randint(x-50, x+50)
        by = y
        bz = random.randint(z-50, z+50)
        if mc.getHeight(bx, bz) > -50:
            by = mc.getHeight(bx, bz)
        if mc.getBlock(bx, by, bz) in [block.GLASS.id, block.GLASS_PANE.id]:
            mc.setBlock(bx, by, bz, block.AIR.id)
            continue
        mc.setBlock(bx, by, bz, block.GRAVEL.id)
        mc.setBlocks(bx, by-1, bz, bx, -60, bz, block.AIR.id)

    mc.postToChat('Sinkhole!')
    blks = []
    y = mc.getHeight(x, z)
    xdist = random.randint(1, 20)
    for bx in range(-xdist, xdist+1):
        zdist = random.randint(1, 20)
        for bz in range(-zdist, zdist+1):
            blks.append([x+bx, z+bz])
    for blk in blks:
        mc.setBlocks(blk[0], mc.getHeight(blk[0], blk[1]), blk[1], blk[0], -60, blk[1],
block.AIR.id)
        mc.setBlocks(blk[0], -55, blk[1], blk[0], -60, blk[1], block.LAVA.id)
    for blk in blks:
        mc.setBlock(blk[0], y, blk[1], block.GRAVEL.id)

disasters = [sinkhole]
def main(disasters, mc):
    baseed = random.randint(1, 10000)
    mc.postToChat('Press button for disaster!')
    while True:
        if GPIO.input(23) == GPIO.HIGH: #Look for button press
            t = random.randint(5, 60)
            t = 5
            time.sleep(t)
            random.seed(baseed + t)
```

Co-funded by the Erasmus+ Programme of the European Union

```
            baseed = random.randint(1, 10000)
            random.shuffle(disasters)
            disaster = random.choice(disasters)
            ppos = mc.player.getTilePos()
            disaster(ppos.x, ppos.z)
try:
    import _thread as thread
except ImportError:
    import thread
thread.start_new_thread(main, (disasters, mc))
```

1.5 Volcano eruption

1.5.1 Equipment and scenario

This activity will present a volcano eruption to the Minecraft Pi world. Once again, students will connect a button to the Raspberry Pi GPIO and program it to create a volcanic eruption after a press.

For this activity you should use the following electronics and peripherals:

1. 1 x Breadboard
2. 1 x Push button with button cap
3. 1 x 220 Ohm Resistor
4. 2 x Male-to-Female Jumper cables
5. 1 x Carton construct

After everything is connected, the students should load the “Fishing Town” map and the Python script named “volcano.py” which can be found in the desktop of each STEM4CLIM8 console.

Then each time the button is pressed, a volcano will appear in the Minecraft Pi world which will erupt, and lava will start flowing towards the fishing town. After a while, the lave will become stone, similar to a real-life volcanic eruption. Buildings and other blocks will disappear, and trees and grass will catch on fire.

A follow-up discussion with your students on the impact of buildings and over population of cities should follow after the activity since physical disasters have no limits and may threaten our lives and status of living.

1.5.2 Electronic circuit and physical block

Connecting a push button to the Raspberry Pi’s GPIO is a very simple process as depicted in Figure 7 below:

1. Connect the right side of the push button to input pin 16 (GPIO23) using a male-to-female jumper cable.
2. Connect the left side of the push button to the 220 Ohm resistor.
3. Connect the other side of the resistor to 3.3V pin 1.
4. You circuit is ready.

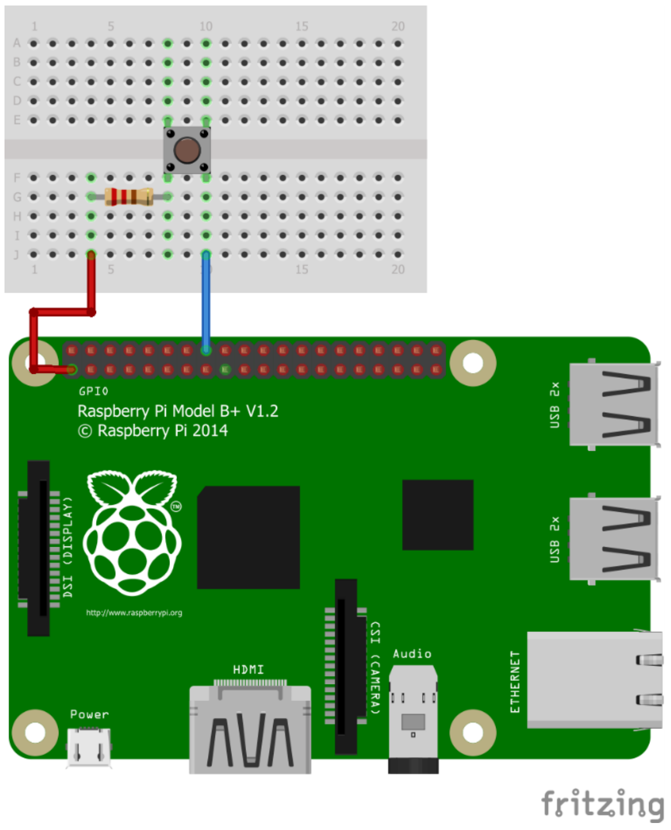

Figure 7 Connecting a push-button to the GPIO using a breadboard and two jumper cables.

Understanding the different GPIO pins, you should look at Figure 8 which describes each pin's properties.

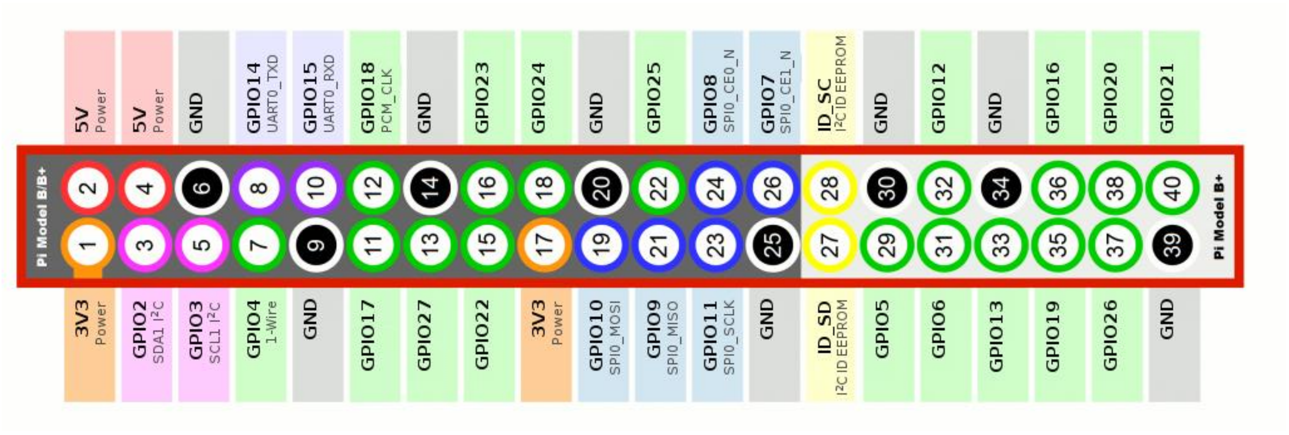

Figure 8 Raspberry Pi GPIO Pins

After creating the circuit, you need to assembly a DIY carton construct and place the circuit in it. For that purpose, you need to use the following printable image. For a better experience, make sure to print the construct in A4 white carton paper and then cut the specified edges.

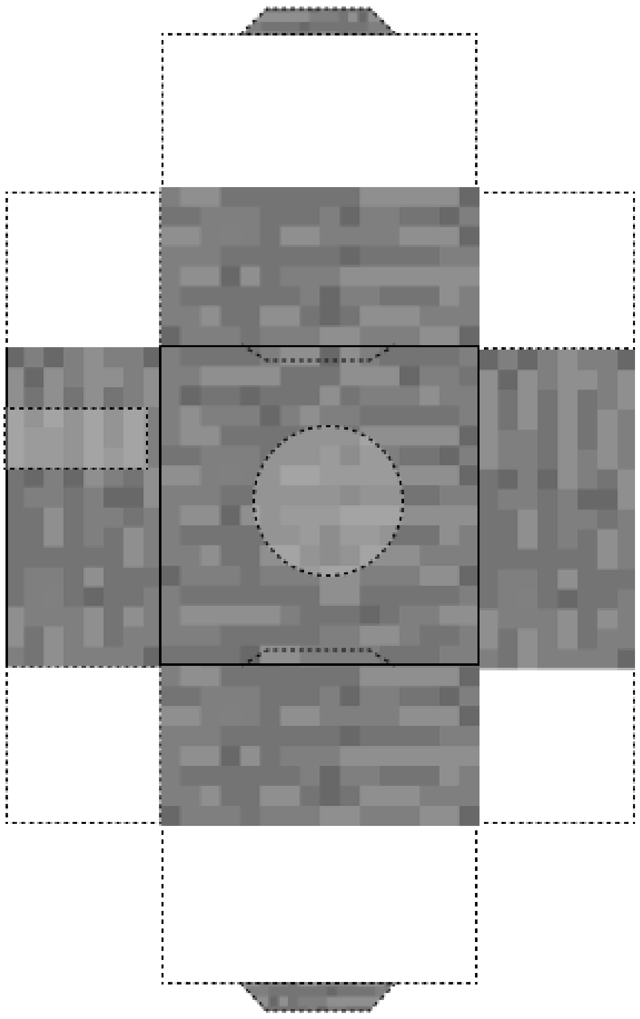

Figure 9 Printable carton construct for physical computing block with button.

1.5.3 Python programming

Simply open Thonny Python on your Raspberry Pi desktop, then load the script called "volcano.py" which can be found on the desktop or copy and paste the script directly to the Python compiler of your choice. If you decide to connect the push-button to a different GPIO pin, please make sure to change the GPIO number in the script below (lines 9 and 13).

```
import RPi.GPIO as GPIO
import mcpi.minecraft as minecraft
import mcpi.block as block
import random, time

mc = minecraft.Minecraft.create()

GPIO.setmode(GPIO.BCM)
GPIO.setup(23, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)

mc.postToChat('Press button for eruption!')
while True:
    if GPIO.input(23) == GPIO.HIGH: #Look for button press
```

```
        mc.postToChat("Eruption")
        x, y, z = mc.player.getPos()
        mc.setBlocks(x,y,z,x+20,y+15,z+20,block.AIR)

        height = 10
        center = x+15, y, z+15

        for i in range(height): #Create the base
            size = height - i
            mc.setBlocks(center[0] - size, center[1] + i, center[2] - size, center[0] +
size, center[1] + i, center[2] + size, block.STONE)

        while True:
            mc.setBlock(center[0], center[1]+height, center[2], block.LAVA_FLOWING)
            time.sleep(1)
```

1.6 Heatwave

1.6.1 Equipment and scenario

This activity will present a heatwave to the Minecraft Pi world. Students will connect an ultrasonic sensor to the GPIO and program it to create a heatwave when an object moves closer to the sensor.

The activity will use the following equipment:

1. 1 x Breadboard
2. 1 x HC-SR04 Ultrasonic Sensor
3. 3 x 1k Ohm Resistors
4. 4 x Male-to-Female Jumper cables
5. 1 x Carton construct

After everything is connected, the students should load the "Fishing Town" map and the Python script named "heatwave.py" which can be found in the desktop of each STEM4CLIM8 console.

As closer the "sun" is coming to the earth, the heatwave phenomenon will appear causing the death of trees, plants and grass, similar to real-life catastrophe.

A follow up discussion with your students on the phenomenon of the rise of the temperature and the negative results to the physical environment should follow after the activity.

1.6.2 Electronic circuit and physical block

First thing you need to do is to create the circuit and connect the HC-SR04 sensor to the GPIO pins of the Raspberry Pi. Before proceeding, you need to turn off your Raspberry Pi and unplug it. The complete circuit is in the schematic diagram that follows.

The HC-SR04 ultrasonic distance sensor comes with 4 pins: power (VCC), trigger (TRIG), echo (ECHO), and ground (GND). The power pin will be connected to the Raspberry Pi's 5V pin, trigger will be assigned to a GPIO pin as output (pin 4), echo will be assigned to a GPIO pin as input (pin 18), and ground will be connected to a ground pin on the Raspberry Pi GPIO.

The program will operate in a way that whenever an object moves closer to the sensor, the sensor emits an ultrasound pulse and the program keeps timing how long it takes to receive the echo back (i.e. how much time passed while the sound waves were emitted, hit the object in front of the sensor, bounced back, and came back to the sensor).

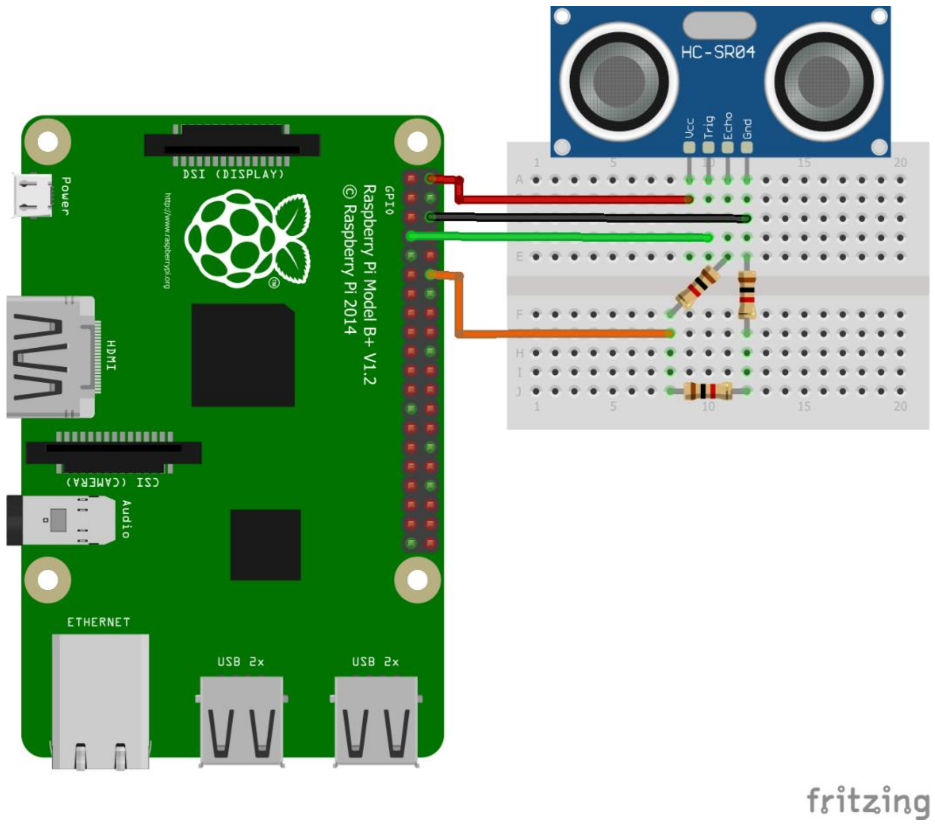

Figure 10 Schematic diagram of circuit with distance sensor connected to GPIO pins.

Understanding the different GPIO pins, you should look at Figure 11 which describes each pin's properties.

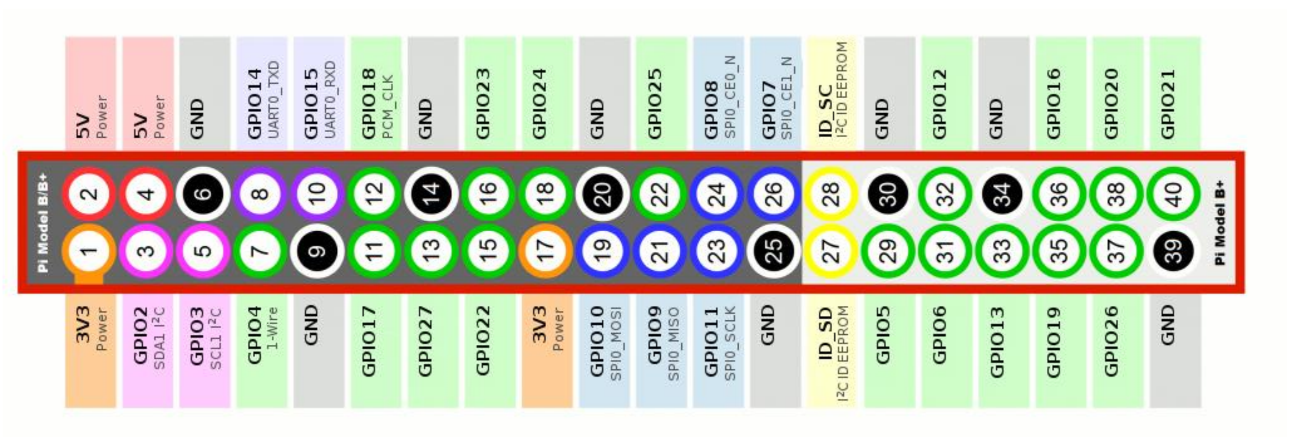

Figure 11 Raspberry Pi GPIO Pins

After creating the circuit, you need to assembly a DIY carton construct and place the circuit in it. For that purpose, you need to use the following printable image. For a better experience, make sure to print the construct in A4 white carton paper and then cut the specified edges.

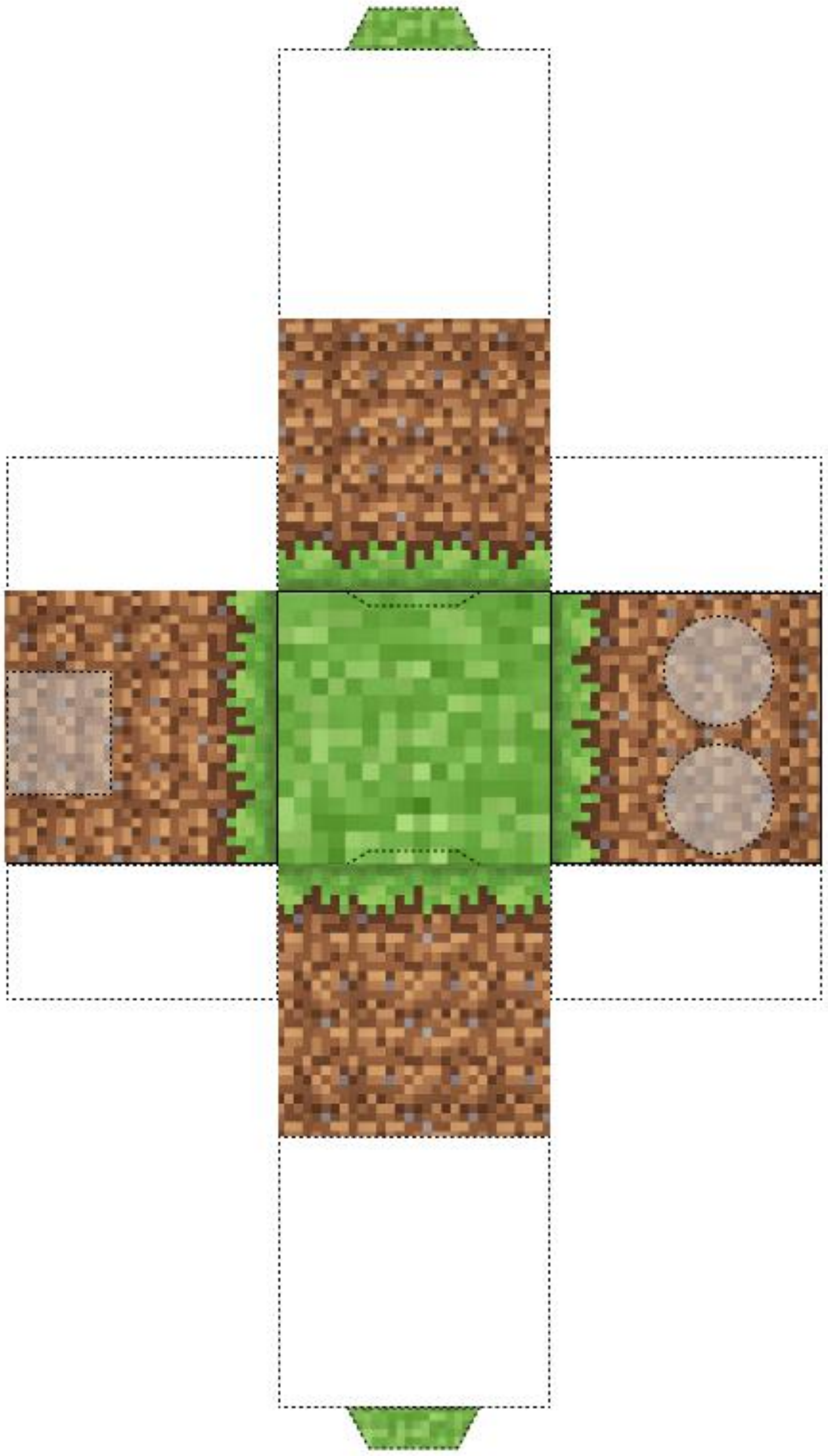
Figure 12 Printable carton construct for physical computing block with distance sensor.

The DIY carton construct represents the "Earth". You will also need another object of similar size which will represent the "Sun". Included in the STEM4CLIM8 package, you will find a 3D printed sphere which you can use. In addition, you can 3D print your own "sun" by using the relevant *.stl* files that are included in the Raspberry Pi desktop.

1.6.3 Python programming

When you are done with the circuit, you can turn on the Raspberry Pi and start Thonny Python, then load the script called "heatwave.py" which can be found on the desktop or copy and paste the script directly to the Python compiler of your choice. In short, when you run the program, the distance HC-SR04 sensor will start emitting ultrasound burst. Each time you move closer the "sun" to the sensor ("Earth") the heatwave phenomenon will happen in the Minecraft Pi world. You will see trees, flowers and plants catching fire and becoming ash.

```
import RPi.GPIO as GPIO
import mcpi.minecraft as minc
import mcpi.block as block
mc = minc.Minecraft.create()
import random, time

GPIO.setwarnings(False)
GPIO.cleanup()
GPIO.setmode(GPIO.BCM)
GPIO.setup(23, GPIO.IN, pull_up_down=GPIO.PUD_DOWN) #pin for push button

#defining TRIG and ECHO pins
TRIG = 4
ECHO = 18
GPIO.setup(TRIG, GPIO.OUT)
GPIO.setup(ECHO, GPIO.IN)

def heatwave(x, z):
    mc.postToChat('Heatwave!')
    y = mc.getHeight(x, z)
    endtime = time.time() + random.randint(50, 90)
    while time.time() < endtime:
        blkid = block.AIR.id
        while blkid == block.AIR.id:
            bx = random.randint(x-10, x+10)
            by = random.randint(y, y+10)
            bz = random.randint(z-10, z+10)
            blkid = mc.getBlockWithData(bx, by, bz).id
        blk = blkid
        blkd = mc.getBlockWithData(bx, by, bz).data
        if blkid == block.GRASS.id:
            blk = block.DIRT.id
            blkd = 0
        elif blkid in [block.WATER.id, block.WATER_FLOWING.id, block.WATER_STATIONARY.id]:
            blk = block.WATER.id
            blkd = 1
        elif blkid == block.LEAVES.id:
            blk = block.COBWEB.id
            blkd = 0
        elif blkid == block.WOOD.id:
            blk = block.LAVA_STATIONARY.id
            blkd = 1
        mc.setBlock(bx, by, bz, blk, blkd)

disasters = [heatwave]
def main(disasters, mc):
    baseed = random.randint(1, 10000)
    mc.postToChat('Push button for sensor to work!')
    mc.postToChat('Move "sun" closer to sensor, and wait for Heatwave!')
    #emit a burst of ultrasound
    GPIO.output(TRIG, True)
    time.sleep(0.00001)
    GPIO.output(TRIG, False)
    StartTime = time.time()
    StopTime = time.time()
    while True:
        if GPIO.input(23) == GPIO.HIGH:
            while GPIO.input(ECHO) == False:
                StartTime = time.time()
            while GPIO.input(ECHO) == True:
                StopTime = time.time()
            TimeElapsed = StopTime - StartTime
            distance = (TimeElapsed * 34300) / 2
            while distance < 15:
                print (distance)
                t = random.randint(5, 120)
                t = 5
                time.sleep(t)
                random.seed(baseed + t)
                baseed = random.randint(1, 10000)
                random.shuffle(disasters)
                disaster = random.choice(disasters)
                ppos = mc.player.getTilePos()
                disaster(ppos.x, ppos.z)
try:
    import _thread as thread
except ImportError:
    import thread
thread.start_new_thread(main, (disasters, mc))
```

1.7 Flood

1.7.1 Equipment and scenario

This activity will present a flood to the Minecraft Pi world. Students will connect a raindrop sensor to the GPIO and program it to create a flood each time the sensor touched water.

The idea is to create a physical computing block using the following equipment:

1. 1 x Breadboard
2. 1 x Raindrops Detection Sensor
3. 3 x Male-to-Female Jumper cables
4. 2 x Female-to-Female Jumper cables
5. 1 x Carton construct

After everything is connected, the students should load the “Fishing Town” map and the Python script named “flood.py” which can be found in the desktop of each STEM4CLIM8 console.

Then each time water is dropped on the sensor, a flood should happen in the Minecraft Pi world causing the drowning of everything. Heavy rain will be referenced with a message in the Minecraft world which will create a Geyser (flood).

A follow-up discussion with your students on the wrong practices and the reasons the flood occurred after the heavy rain should follow after the activity.

1.7.2 Electronic circuit and physical block

Connecting a raindrop sensor to the Raspberry Pi's GPIO is a relatively easy process which is shown in Figure 13 below:

1. The raindrop sensor control board has 4 pins which can be connected to a GPIO, namely, voltage (VCC), ground (GND), digital output (DO) and analogue output (AO).
2. You need to connect the following through a breadboard or directly on the GPIO:
    a. VCC to 5V (pin 2)
    b. GND to GND (pin 4)
    c. DO to GPIO23 (pin16)
3. Then, you need to connect the sensor detection board to the control board by using two jumper cables, connecting the following:
    a. The + side of the control board to the + side of the detection board.
    b. The – side of the control board to the – side of the detection board.
4. Your circuit is ready.

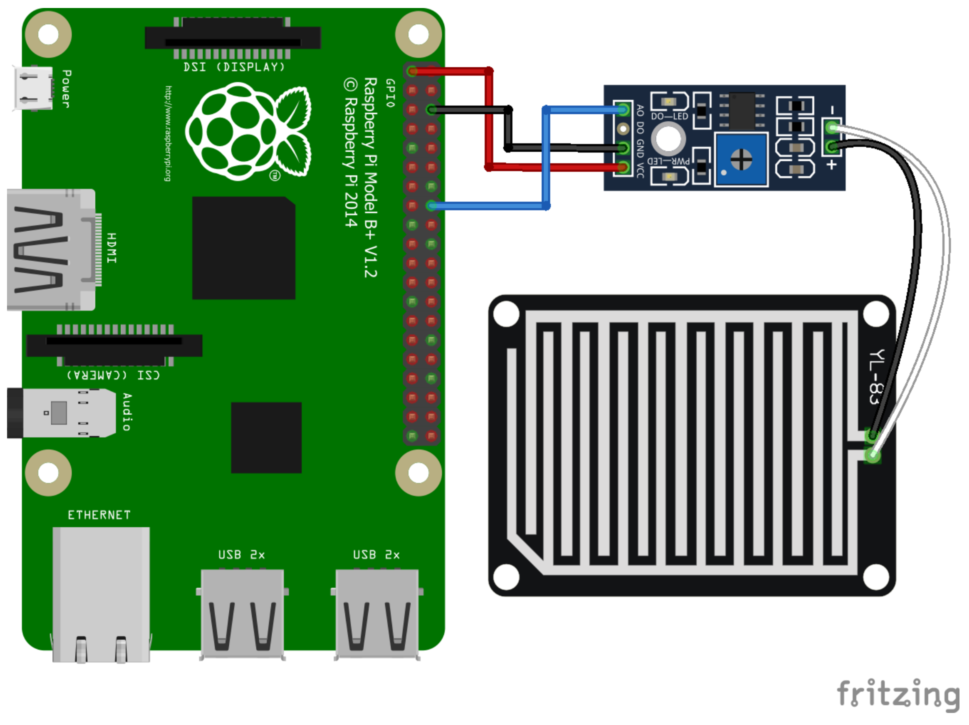

Figure 13 Schematic of raindrop sensor connected to Raspberry Pi's GPIO pins.

Understanding the different GPIO pins, you should look at Figure 14 which describes each pin's properties.

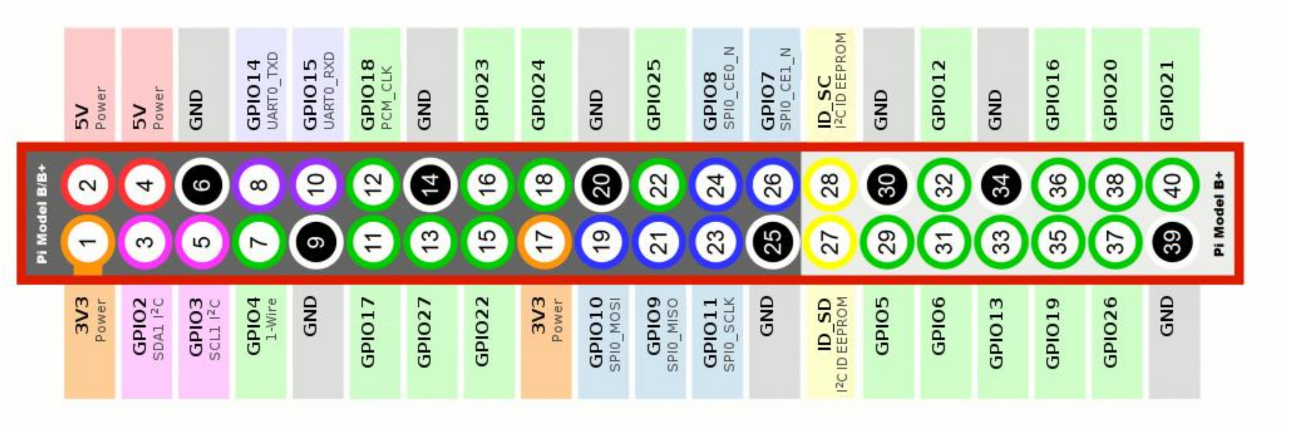

Figure 14 Raspberry Pi GPIO Pins

A raindrop sensor can send both analogues and digital outputs to the controlling device. In this case, the Raspberry Pi computer only reads digital signals. So, for this activity, only digital output is needed since the program will run each time the sensor detects a raindrop, causing a flood in the Minecraft Pi world.

After creating the circuit, you need to assembly a DIY carton construct and place the circuit in it. For that purpose, you need to use the following printable image. For a better experience, make sure to print the construct in A4 white carton paper and then cut the specified edges.

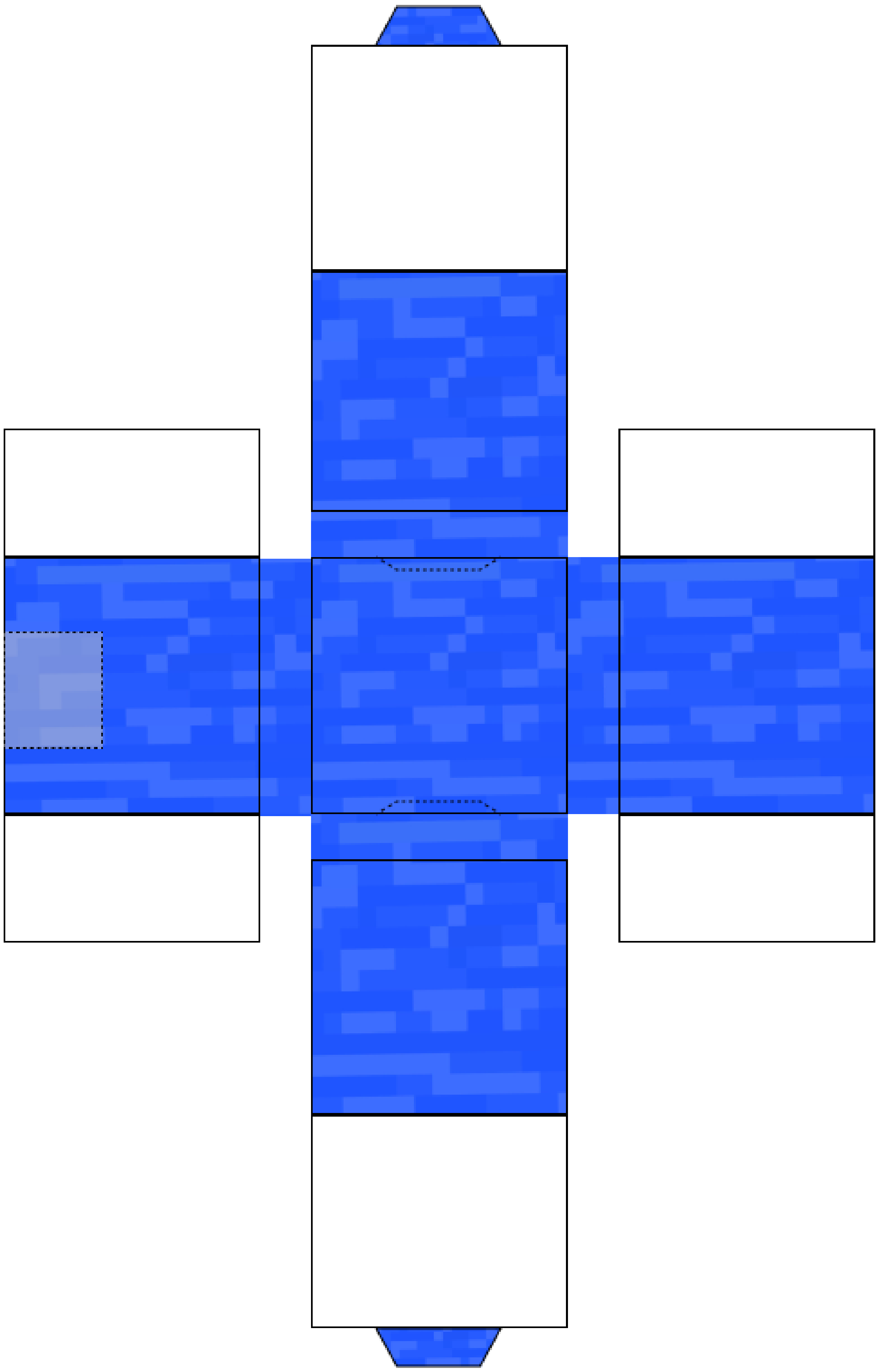

Figure 15 Printable carton construct for physical computing block with raindrop sensor.

1.7.3 Python programming

```
import RPi.GPIO as GPIO
import mcpi.minecraft as minc
import mcpi.block as block
mc = minc.Minecraft.create()
import random, time

GPIO.setmode(GPIO.BCM)
GPIO.setup(23, GPIO.IN)

def geyser(x, z):
    mc.postToChat('Flood!')
    y = mc.getHeight(x, z)
    mc.setBlocks(x-2, y+5, z-2, x+2, -60, z+2, block.WATER.id)
    time.sleep(25)
    mc.setBlocks(x-2, y+5, z-2, x+2, -60, z+2, block.AIR.id)

disasters = [geyser]
def main(disasters, mc):
    baseed = random.randint(1, 10000)
    mc.postToChat('Drop water on sensor for disaster!')
    while True:
        if GPIO.input(23) == GPIO.LOW: #Look for raindrop sensor activation
            t = random.randint(5, 120)
            t = 5
            time.sleep(t)
            random.seed(baseed + t)
            baseed = random.randint(1, 10000)
            random.shuffle(disasters)
            disaster = random.choice(disasters)
            ppos = mc.player.getTilePos()
            disaster(ppos.x, ppos.z)
try:
    import _thread as thread
except ImportError:
    import thread
thread.start_new_thread(main, (disasters, mc))
```

1.7 Conclusion

The use of these activities will be for the kids to experience the physical disasters and to understand the role of human activities in worsen these phenomena, such as climate change, etc.

References

Minecraft Pi Edition: SURVIVAL mode-ish. Retrieved from https://teachwithict.weebly.com/computing-blog/minecraft-pi-edition-survival-mode-ish#sthash.S6XeUUDH.dpbs Last access: 19/11/2021

How to use Rain Sensor Module with Arduino & Raspberry Pi by SunFounder Maker Education. Retrieved from https://www.youtube.com/watch?v=Xnisf0GP9bA&t=203s Last access: 19/11/2021

Using a Raspberry Pi distance sensor (ultrasonic sensor HC-SR04). Retrieved from https://tutorials-raspberrypi.com/raspberry-pi-ultrasonic-sensor-hc-sr04/ Last access: 19/11/2021