CrowPi
by Elecrow

CrowPi with MineCraft Pi Edition

Quick Start Guide(V2.1 2019.11）

CrowPi with MineCraft Pi Edition

- Introduction - Minecraft PiEdition
- Introduction - What you willneed
- Introduction - RunningMinecraft
- Introduction - Playing Multiplayer with moreCrowPi’s
- Lesson 1 - Minecraft Pi editionAPI
- Lesson 2 - Using the Pythoninterface
- Lesson 3 - Finding your location in thegame
- Lesson 4 - Teleporting theplayer
- Lesson 5 - Generatingblocks
- Lesson 6 - Dropping blocks as youwalk
- Lesson 7 - Playing with TNTblocks
- Lesson 8 - Playing with burningLava
- Lesson 9 - Getting player position on the LCDscreen
- Lesson 10 - Teleporting player on touchpress
- Lesson 11 - Building house using pythoncode
- Lesson 12 - TNT Buzzerdetector
- Lesson 13 - Generating blocks usingNFC

Introduction

What is Minecraft Pi Edition?

Minecraft Pi Edition is a special edition of minecraft crafted for the Raspberry Pi.

The version is based on the “Minecraft pocket edition” with minimal features that allows the raspberry pi to run smoothly.

From the official Minecraft website:

“Have you ever thought about learning to program? Where would you begin? How much would it cost? What would you need to get things moving?

The Raspberry Pi is a credit card-sized computer that’s a great starting point. It’s cheap, capable, and approachable for newbie programmers. And we’ve made a FREE version of Minecraft just for it!

It comes with a revised feature set and support for multiple programming languages.

You can start by building structures in the traditional Minecraft way, but once you’ve got to grips with the in-game features, there’s opportunity to break open the code and use programming language to manipulate things in the game world. You’ll be learning new skills through Minecraft! Wow!”

Minecraft Pi Edition combined with CrowPi is the perfect way to go if you are interested in learning coding and micro electronics using fun and simple steps!

Introduction

What will you need to get things going?

For our tutorials and lessons you'd need the followings:

*** CrowPi (any version from basic to advanced willwork!**

*** Raspberry Pi Zero / 2 / 3 / 4 /5 (connected to the CrowPi)**

*** Mouse and keyboard (can be wired orwireless)**

This version of Minecraft comes with the Raspberry Pi Raspbian built in so we don't need to install any additional software.

Wifi or bluetooth are not required unless you're connecting wireless keyboard and mouse, and if an item is controlled by bluetooth, you might need to enable it and connect it the right way.

Introduction

Running minecraft and getting things started

If you navigate to the upper menu where the “Raspberry Pi” logo is, you’ll find a navigation bar called “Games”.

Under Games, you’ll be able to see 2 options: Minecraft and Python games.

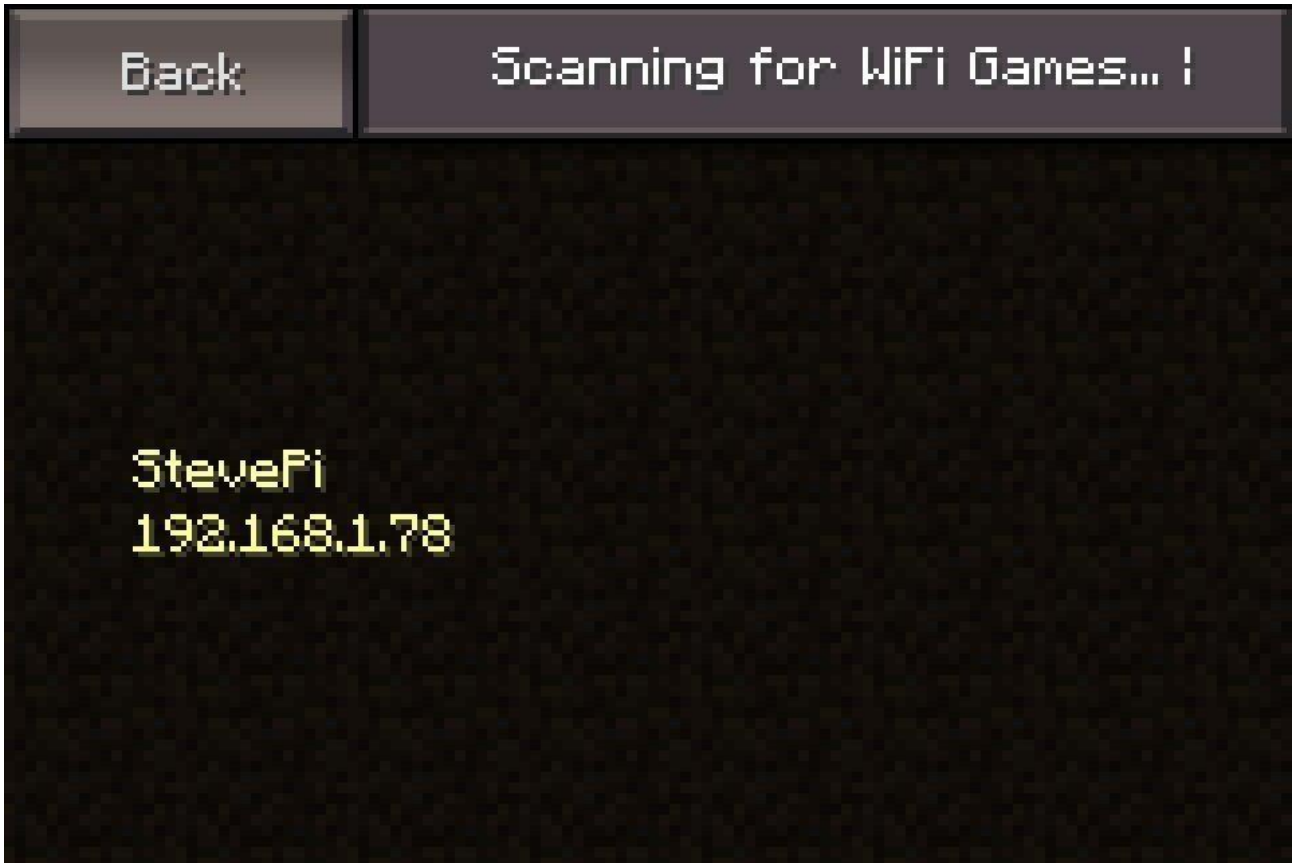
Left click on Minecraft and the game will open and should be ready to play!

Next thing will be to prepare our python interface going and we’ll be ready to go!

Note: If you’re curious, you should check python games out. It includes multiple fun games all coded in python, it’s a great way to spend good time with your CrowPi and your friends!

Introduction

Playing multiplayer with more CrowPi’s



It’s actually pretty interesting, Minecraft is multiplayer supported!

If you or one of your friends have a spare CrowPi, you can play together!

Study, code and learn both easily and quickly.

The way to do it is one of you should create a new “world” in Minecraft and then under multiplayer tab the other CrowPi will be able to discover the “server” and join.

In order to do this, you both should be on the same network.

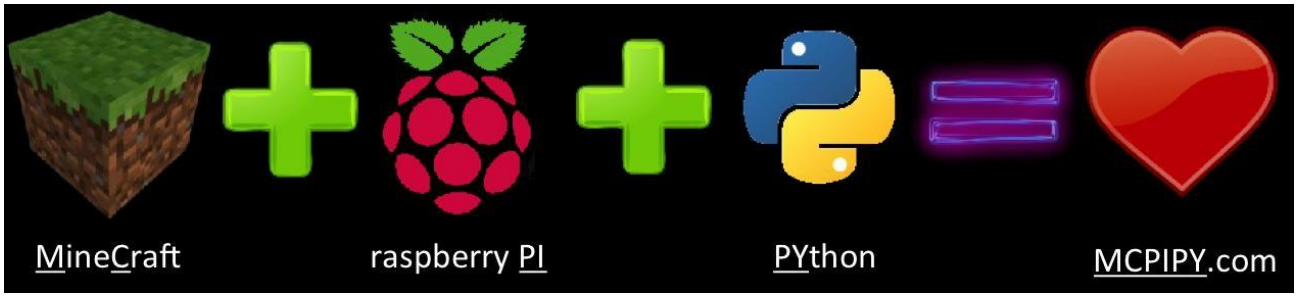
What’s more awesome? It also supports the Minecraft Pocket edition which means if your friends have iPhone / Android / Tablet or even a game console they can join in on your CrowPi game!

That’s right!

Make sure to let your friends know you have your own Minecraft idea using the CrowPi and let them join you to craft great things together.

Lesson 1

Using the Minecraft Pi Edition API



Minecraft Pi edition has an API to control the Minecraft Game.

API stands for “Application programming interface” and by having Minecraft API it allows us to control the game using Python programming language! Isn’t that awesome?

The API called “MCPIPY” MC stands for MineCraft, PI from the Pi and PY from Python.

We will use MCPIPY (comes built in in the Raspbian) in order to control the game during the lesson and make some pretty cool things.

Want an example? Let’s take a look:

```
from mcpi.minecraft import Minecraft

mc = Minecraft.create()

mc.postToChat("Hello world")
```

This is a very simple usage for our API using Python programming language.

First, we import minecraft by using the “mc = MineCraft.create()” command, now “mc” has the minecraft game object in it which we’ll use to control the game

Then we give the game a command:

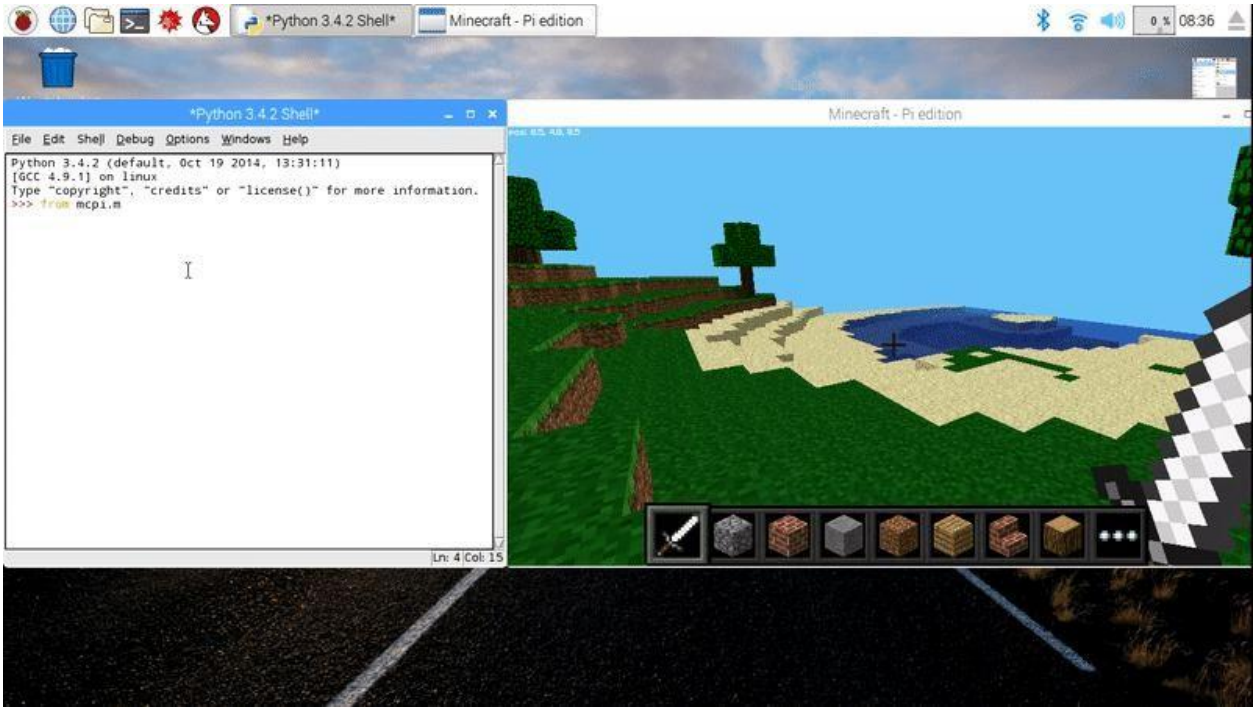
mc.postToChat(“Hello world”)

Can you guess what this command does? No?

Let’s learn how to use the python command interface so you could try and see by yourself :)

Lesson 2

Using the Python interface to control the game



Go to the menu and select “Python” from the application menu,

This will open you an interface which will allow you to code directly into the game.

You can try to type line by line the first example of “hello world” we showed last lesson and see what happens.

If everything goes right, you should see “hello world” chat pops out! Exciting isn’t it?

Using the Python shell we’ll be able to prototype quickly and try different commands and things. If you want to go more advanced, create your own file or execute the examples we’ll give later in order to see what happens.

The Minecraft Pi Edition api (MCPIPY) is pretty big and allows various functionalities. During the lessons we’ll go through many of them.

If you’d like to read the API by yourself you can find it at the following link:

https://www.stuffaboutcode.com/p/minecraft-api-reference.html

Lesson 3

Finding your location in the game

Finding your position in the game is necessary.

Later on, in the game, we'll learn how to generate blocks and do multiple actions, in order to dothose

```
pos = mc.player.getPos()
```

things you need to know **where** to do them.

We'll need to find our position in the game for that purpose. How we'll do that? Let's find out.

That's it. Now "pos" object contains your position data which you can print or use for your own

```
x, y, z = mc.player.getPos()
```

stuff. But there is one more thing we can do …

Now we have different coordinates separated. Now x, y, and z contain each part of your position coordinates. x and z are the walking directions (forward/back and left/right) and y is up/down.

Lesson 4

Teleporting your player

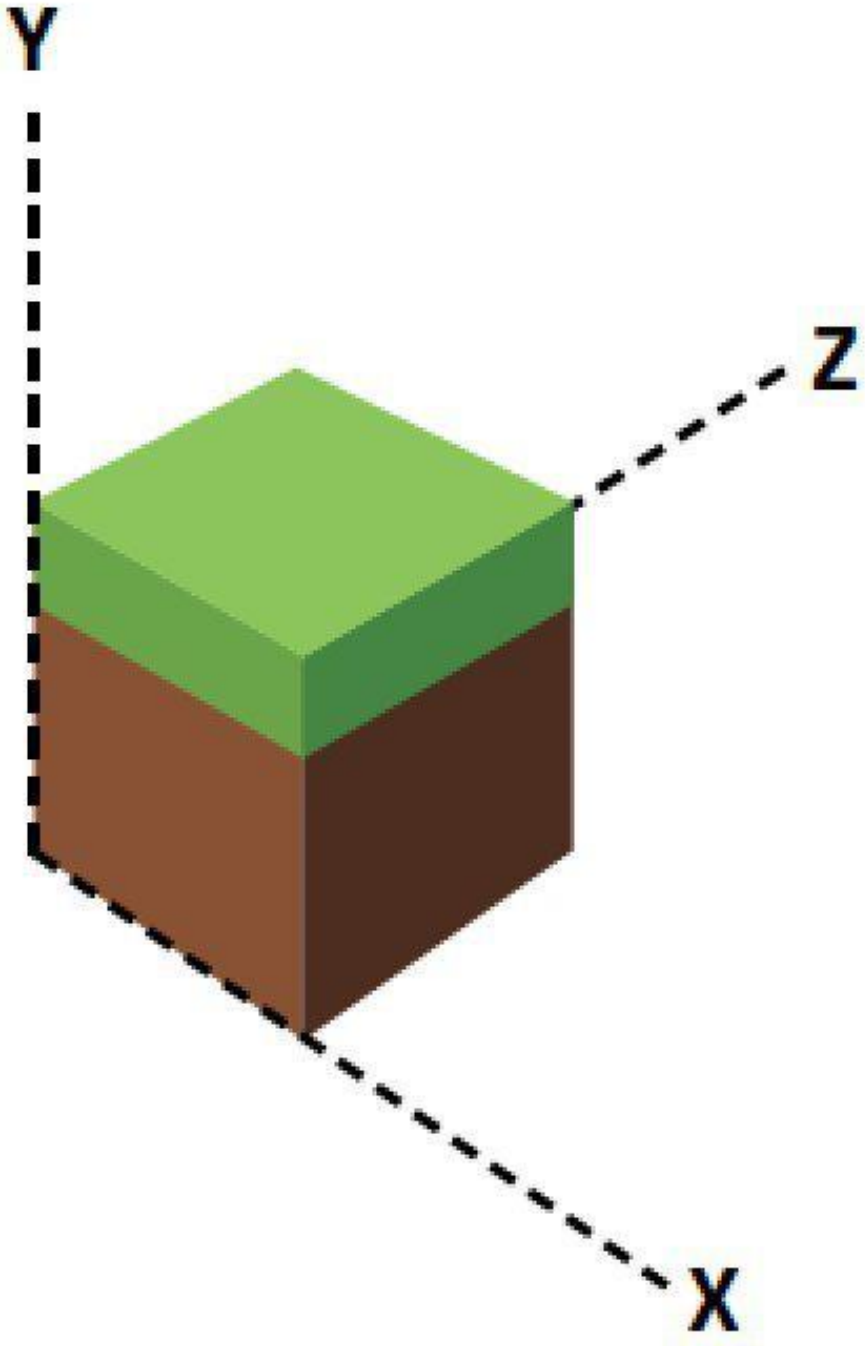
If we can find your current player position, why can't we change it?
This will transport your player to 100 spaces in the air. This will mean you'll teleport to the middle

```
x, y, z = mc.player.getPos()
mc.player.setPos(x, y+100, z)
```

of the sky and fall straight back down to where you started.

Try to change one of the coordinates (x, y or z) and see what happens!
When you go too far, it might be a good idea to save that position,
isn't it?

If you want to better understand how teleportation works, you can use the following image.



As you can see the Y stands for height Z for and X for width on different directions,
While considering where to move or teleport your player, if you're not sure how to calculate the positions you can use and refer to this picture!

Lesson 5

Generating and playing with blocks

Blocks is something we'll use very often during our lessons, we should learn how to generate them!
There's a very simple function called "setBlock()" let's see how it works:

```
x, y, z = mc.player.getPos()
mc.setBlock(x+1, y, z, 1)
```

As you can see, first we must get our location in order to generate the block next to us so we'll be able to find it. next, we use the setBlock() function, giving it our position and the ID of the block.

We write x+1 so we'll be able to generate the block in front of us (+1 one block from coordinate x)
You can try to change it and see what happens!

The number "1" we specified at the end is the ID of the block, ID number 1 stands for stone block but there are multiple block types such as:

Air:0
Stone:1
Grass:2
Dirt: 3
Etcetc…

You can check the MCPIPY API in order to find all the existing blocks IDs.

We can use is setting block as variable, for example:

```
dirt = 3
mc.setBlock(x, y, z, dirt)
```

We keep the ID of the block inside variable and then use the variable to generate it.

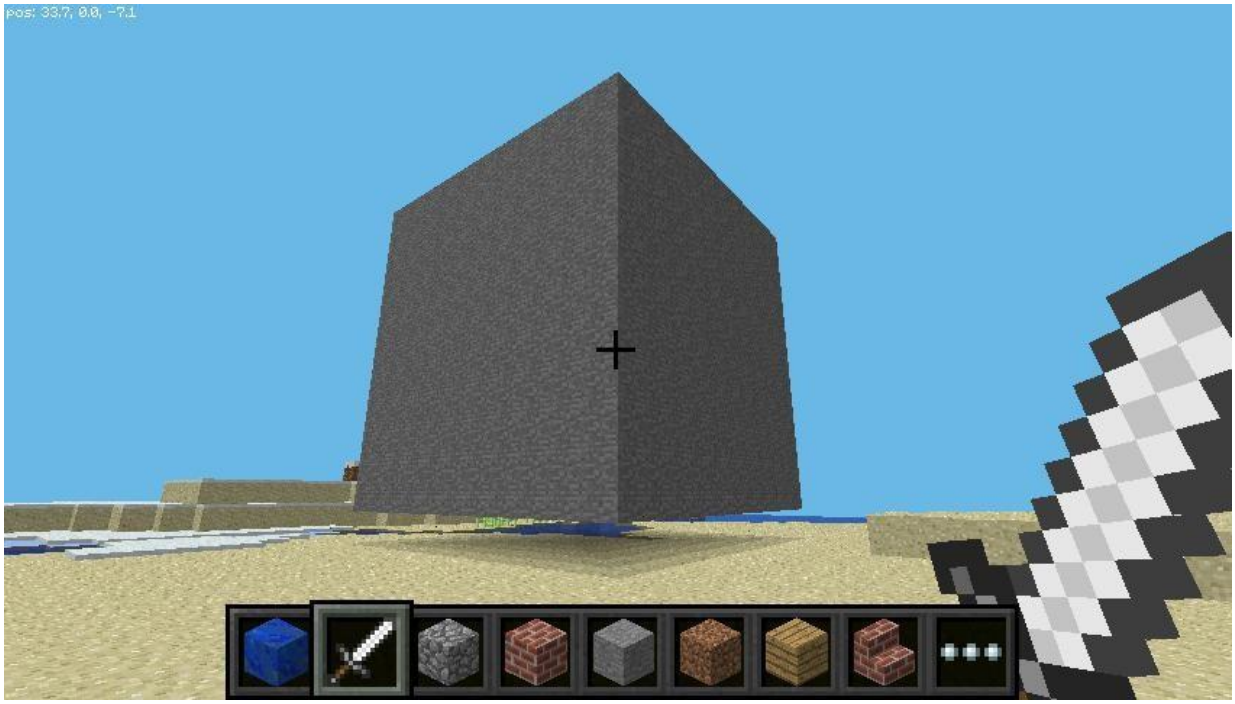
Also, it's important to mention there are some "special" blocks which use extra properties which required in order to generate them for example, wool:
We have multiple of wool such as:

0:White
1: Orange
2:Magenta
3: Light Blue
4: Yellow

```
wool = 35
mc.setBlock(x, y, z, wool, 1)
```

So after setting up the Block ID you'd like to use, you'll need to add additional property to them. If you don't set the extra property, it'll still work! It will go with the default value, in the case of wool that would be 0 which is white.

Other blocks which have extra properties are wood (17): oak, spruce, birch, etc; tall grass (31): shrub, grass, fern; torch (50): pointing east, west, north, south; and more. See the API reference for full details.



We can also set massive block of multiple blocks together!
How we'll do that? Let's take a look

```
stone = 1
x, y, z = mc.player.getPos()
mc.setBlocks(x+1, y+1, z+1, x+11, y+11, z+11, stone)
```

As you can see, first we set the ID of the block which we'd like to use which in our case is stone.

Then we get our position, we set x,y and z +1 to be infront of us, then we set another x, y z which aren't our position properties but the size of the blocks which will be 10x10x10 ! That huge!

The last thing will be to give the function the block id we would like to use which is stone.

You can try different block sizes and different block types but be aware: if you try to generate a massive block for example 1000 or even 10,000 it might crash the game! But why not, try and see if you can find the limits!

Lesson 6

Dropping blocks on the go

After we've learned how to generate multiple types of blocks, why not drop them as we walk in thegame? For our example, let's use some flowers!

```
from mcpi.minecraft import Minecraft
from time import sleep

mc = Minecraft.create()

flower = 38

while True:
    x, y, z = mc.player.getPos()
    mc.setBlock(x, y, z, flower)
    sleep(0.1)
```

As you can see, first we imported the Minecraft module and the time module as well.

Then we've created Minecraft object and flower variables with the flower ID which is 38, while True (forever) we get our position (to get it every time when we walk) and we set flower in our current position, then we wait 100 milliseconds to prevent the game from crashing!

Even if we fly through the air, we'll still be able to see the flowers going around!

What if we want to know on which block we are standing on right now? We can use **getBlock**to

```
x, y, z = mc.player.getPos()  # player position (x, y, z)
this_block = mc.getBlock(x, y, z)  # block ID
print(this_block)
```

find out what type a block is:
This tells you the location of the block you're standing *in* (this will be 0 - an air block). We want to know what type of block we're standing *on*. For this we subtract 1 from the y value and use

```
x, y, z = mc.player.getPos()  # player position (x, y, z)
block_beneath = mc.getBlock(x, y-1, z)  # block ID
print(block_beneath)
```

getBlock() to determine what type of block we're standing on:

This tells us the ID of the block the player is standing on. Test this out by running a loop to print

```
while True:
    x, y, z = mc.player.getPos()
    block_beneath = mc.getBlock(x, y-1, z)
    print(block_beneath)
```

theblock ID of whatever you're currently standing on: Let's say we want to plant flowers in grass only.

Obviously we don't want flowers in the air or in the water … let's try the following:

```
grass = 2
flower = 38

while True:
    x, y, z = mc.player.getPos()  # player position (x, y, z)
    block_beneath = mc.getBlock(x, y-1, z)  # block ID

    if block_beneath == grass:
        mc.setBlock(x, y, z, flower)
    sleep(0.1)
```

We've set grass and flower ids into variables and then in a loop, we'll get our current position and the block beneath us. If the block beneath us equals to grass block - we'll plant a flower!

Isn't that cool?

now, what if there is no grass? How about we put a grass block if we can't find one so we could

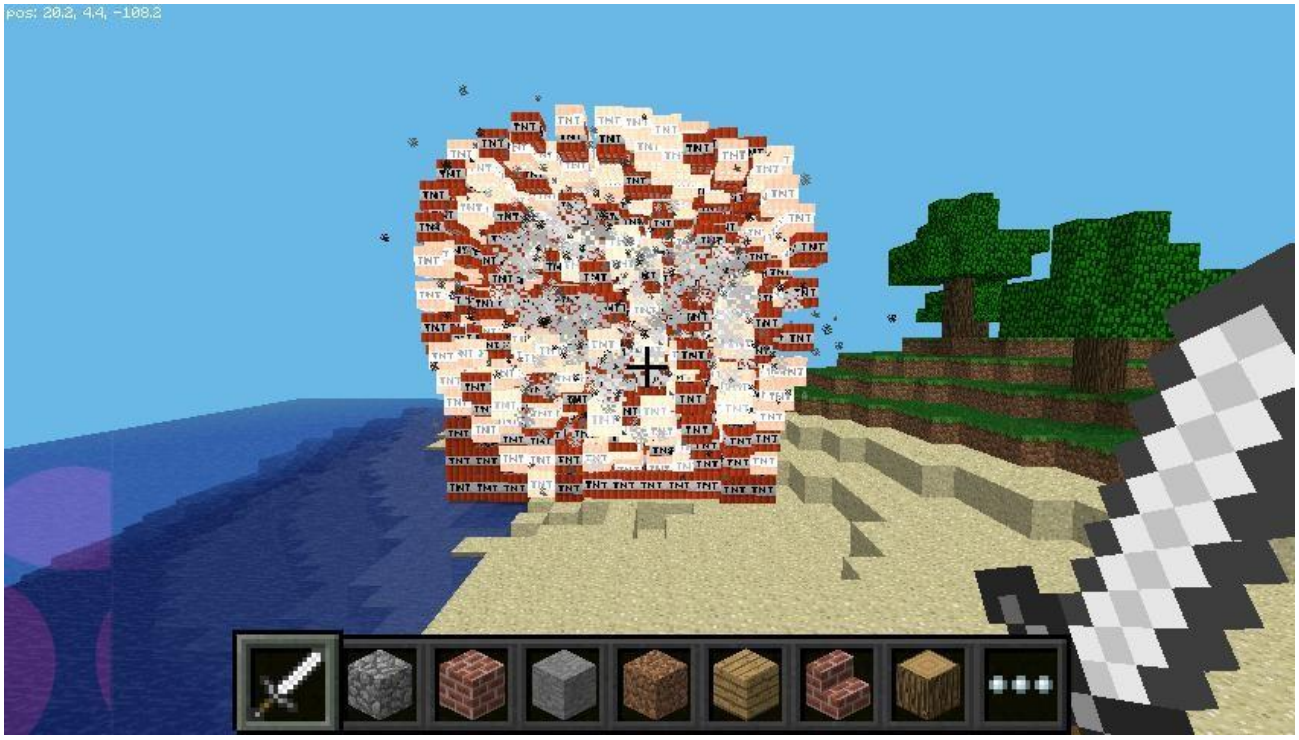
```
if block_beneath == grass:
    mc.setBlock(x, y, z, flower)
else:
    mc.setBlock(x, y-1, z, grass)
```

plant a flower into it:

And that would be our result:

Lesson 7

Playing with TNT blocks



Another pretty cool block in Minecraft is the TNT block, it allows you to make things explode!

Let's see how to generate one:

```
tnt = 46
mc.setBlock(x, y, z, tnt, 1)
```

The TNT block ID is 46, after generating it we'll use x, y and z as our location, tnt as variable and 1 to set "activated" if we don't set 1, it would just stand there like any other block.

By setting 1 it allows you to left click and hit it with your sword and let the show begin, go back

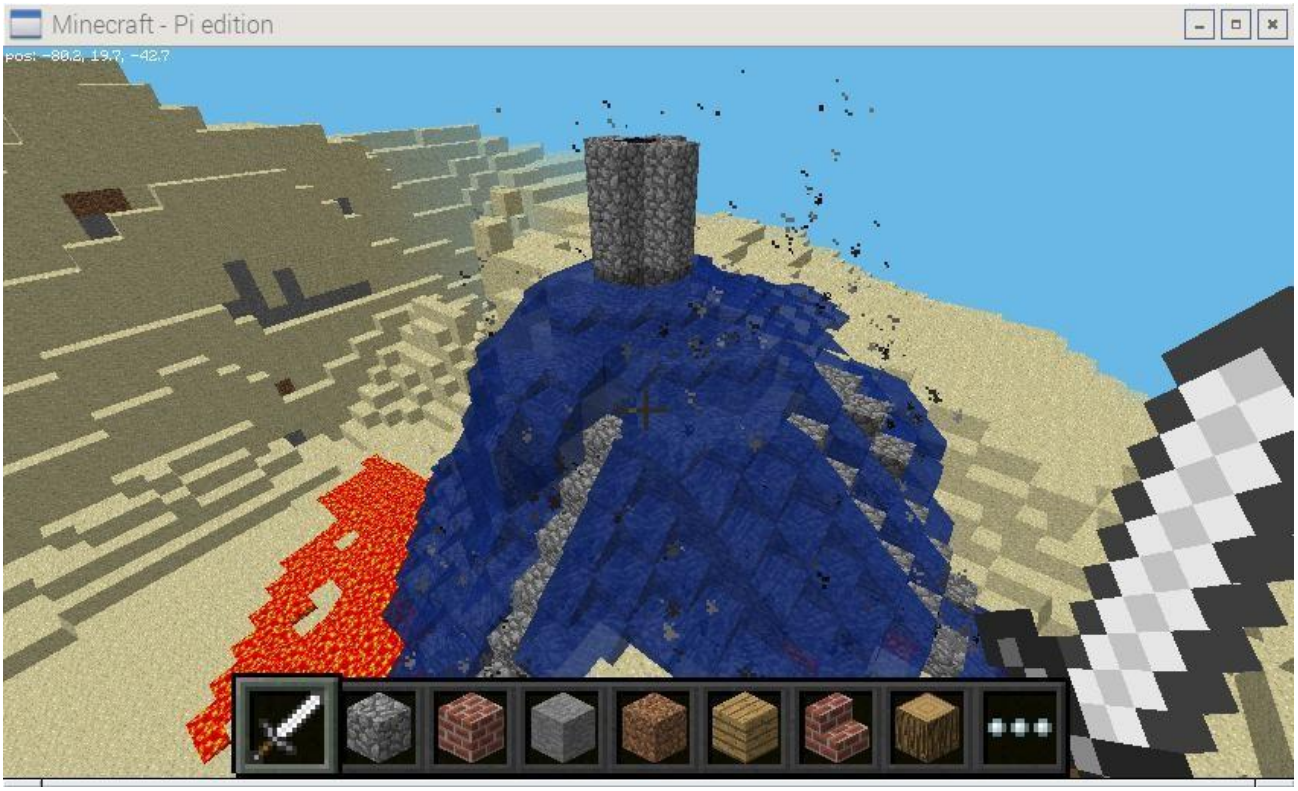
```
tnt = 46
mc.setBlocks(x+1, y+1, z+1, x+11, y+11, z+11, tnt, 1)
```

few blocks and see how it explodes! Try to generate a big block of TNT as well:

Make sure not to generate a block which is too big else it might crash your game

Lesson 8

Playing with flowing, hot burning, Lava.



Another pretty cool block to play with is flowing “Lava”, let’s see how to generate it:

```
from mcpi.minecraft import Minecraft

mc = Minecraft.create()

x, y, z = mc.player.getPos()

lava = 10

mc.setBlock(x+3, y+3, z, lava)
```

The cool thing about lava is when it cools down it becomes a rock which is another different block. Try to generate lava as you walk, the way we learned before and see what happens!

Lesson 9

Getting current player position on LCD

After we've successfully learned how to change and get positions, why not play further?

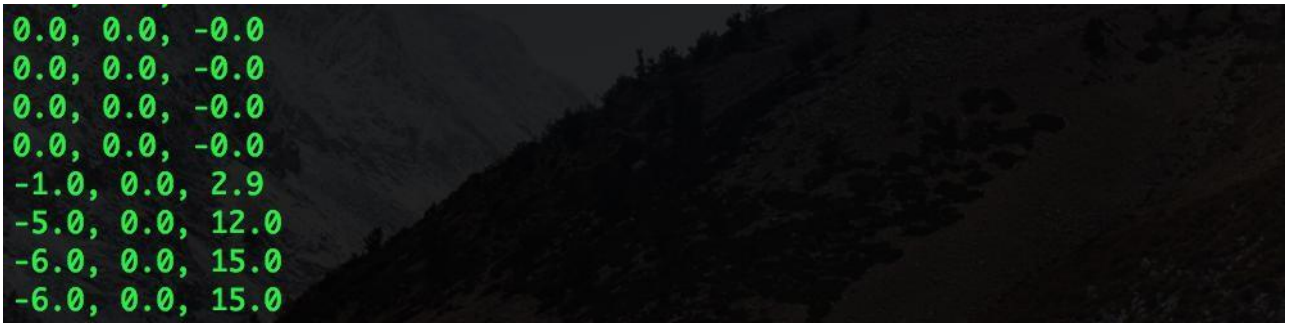
In this example we'll show how to display current player position on top of the CrowPi LCD.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
import time
import HD44780MCP
import MCP230XX
from mcpi.minecraft import Minecraft

# Define Minecraft

mc = Minecraft.create()

#initialize MCP
i2cAddr = 0x21  # MCP23008/17 i2c address
MCP = MCP230XX.MCP230XX('MCP23008', i2cAddr)
#MCP = MCP230XX.MCP230XX('MCP23017', i2cAddr)

# turn on backlight if using Adafruit i2c LCD backpack (uses MCP23008)
blPin = 7 # Back light pin when using Adafruit LCD backpack
MCP.set_mode(blPin, 'output')
MCP.output(blPin, True) # turn backlight on - for Adafruit LCD backpack use

# set 16 character x 2 row LCD screen without rw pin, 4 bit mode
LCD = HD44780MCP.HD44780(MCP, 1, -1, 2, [3,4,5,6], rows = 2, characters = 16, mode = 0, font =



while True:
    # get player position
    x, y, z = mc.player.getPos()
    x, y, z = float(str(x)[:3]),float(str(y)[:3]),float(str(z)[:3])
    pos = (str(x)+", "+str(y)+", "+str(z))
    print(pos)
    LCD.display_string("Position:")
    LCD.set_cursor(2,1)             # move cursor to 2nd row, 1st position
    LCD.display_string(pos)
    time.sleep(1)
    LCD.clear_display()
```

First we get our player position, then we convert our position to float numbers so if we have integers like 1,2,3 it will become 1.0,2.0,3.0 and so on … then we combine them together into a beautiful string so we will be able to show it on top of our LCD … we print our position in the terminal and then show our position on top of the lcd using “lcd.message()” function, then we sleep, clear the lcd screen andrepeat!

Walk through the map to see the updated location keep showing up:

And the terminal printings …



You can download and execute the script from our GitHub page, try it by yourself!

To download run the following commands:

wget https://github.com/Elecrow-RD/CrowPi-for-Raspberry-Pi-5/blob/main/Minecraft/lcd_game_position.py

sudo python lcd_game_position.py

Lesson 10

Teleporting player on touch sensor press

After showing up our current position on the LCD screen … why not change it?

In this example we'll show how we are able to teleport ourselves in the game using the "touch" sensor on GPIO PIN number 17. Take a look at the example:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
import RPi.GPIO as GPIO
from mcpi.minecraft import Minecraft
import time

# create Minecraft Object
mc = Minecraft.create()

# set touch pin
touch_pin = 17
# set gpio mode as GPIO BOARD
GPIO.setmode(GPIO.BCM)
# set as INPUT
GPIO.setup(touch_pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)

while True:
    if GPIO.input(touch_pin) == True: # look for button press
        mc.player.setPos(0, 0, 0) # teleport player
        print("Teleported successfully!")
        time.sleep(0.5) # wait 0.5 seconds
```

First, we set the touch PIN to GPIO 17, then we set the mode to GPIO BCM.

We configure the touch sensor as INPUT (as we press on it), then, forever, we check if the button is pressed. If it was pressed: we teleport the player to position 0, 0, 0 by using the setPos function

We set time.sleep in order to avoid teleporting multiple times while holding our finger on the sensor.

You can change the position and see what happens!

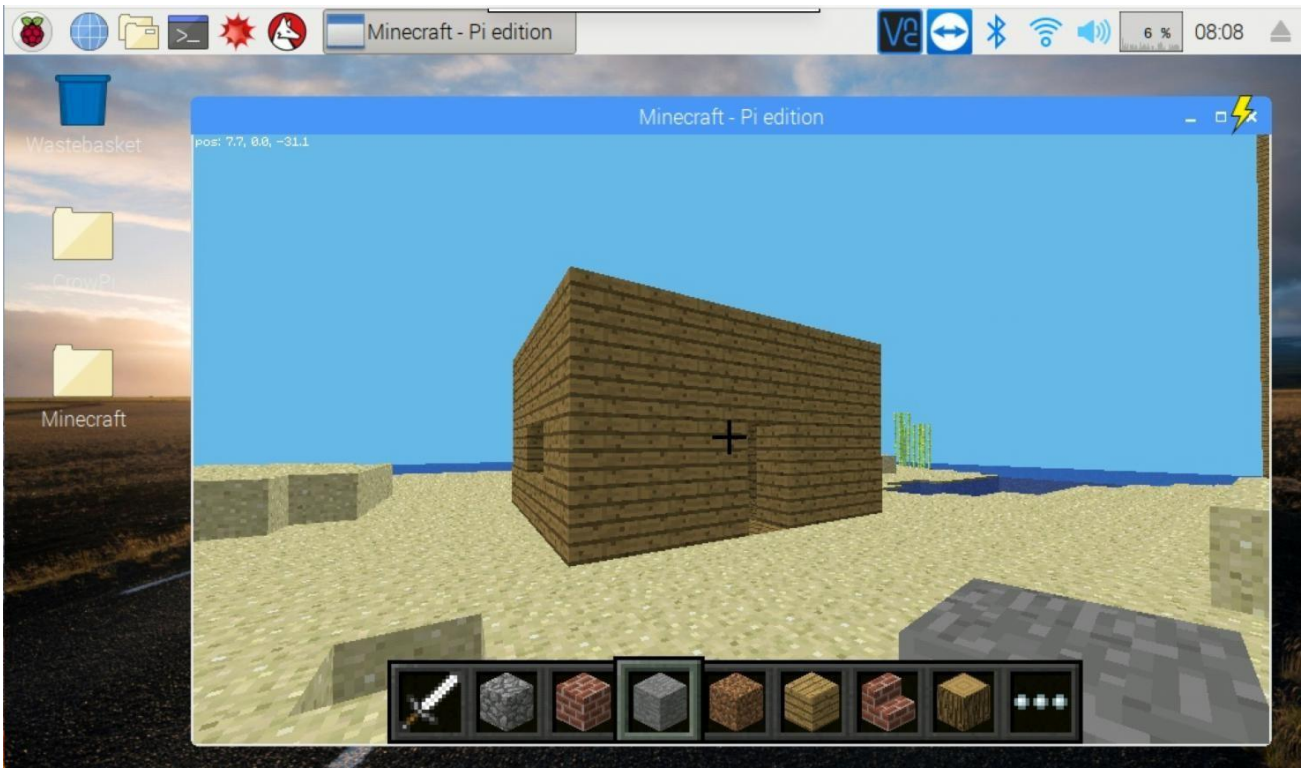
In order to download and execute the script, you can run the following commands:

wget https://github.com/Elecrow-RD/CrowPi-for-Raspberry-Pi-5/blob/main/Minecraft/teleport.py

sudo python teleport.py

Lesson 11

Building a house using a python script



How about building some more advanced things using python and Minecraft?

In this example we'll teach you how to build a house completely from python.

In order for this example to go smoothly, position yourself in a place where there is no water or lava, make sure you are on the ground and not flying around.

Let's take a look at the code and see how to make it!

```
from mcpi.minecraft import Minecraft
import time

mc = Minecraft.create() # create Minecraft Object

x, y, z = mc.player.getPos()
mc.setBlocks(x + 2, y - 1, z + 2, x + 7, y + 3, z + 8, 5) # make shell
mc.setBlocks(x + 3, y, z + 3, x + 6, y + 2, z + 7, 0) # remove inside
mc.setBlocks(x + 2, y, z + 5, x + 2, y + 1, z + 5, 0) # make doorway
mc.setBlocks(x + 4, y + 1, z + 8, x + 5, y + 1, z + 8, 102) # make window 1
mc.setBlocks(x + 4, y + 1, z + 2, x + 5, y + 1, z + 2, 102) # make window 2
mc.setBlocks(x + 7, y + 1, z + 4, x + 7, y + 1, z + 6, 102) # make window 3
```

At first glance the code might seem a bit complicated but hold on … let's try to explain

The first part is to make the shell (the entire house full of wooden blocks)

The second part is to remove the inside of the house so we could enter into it …

The third one is to empty the entry so we could enter! Make it look like a door!

4,5,6 are windows on the sides of the house made of glass! That's right, 102 ID stands for glass, check it out!

And that's how we make a house using python …. incredible.

Try to download and execute the code by yourself and see what happens:

wget https://github.com/Elecrow-RD/CrowPi-for-Raspberry-Pi-5/blob/main/Minecraft/

house.py

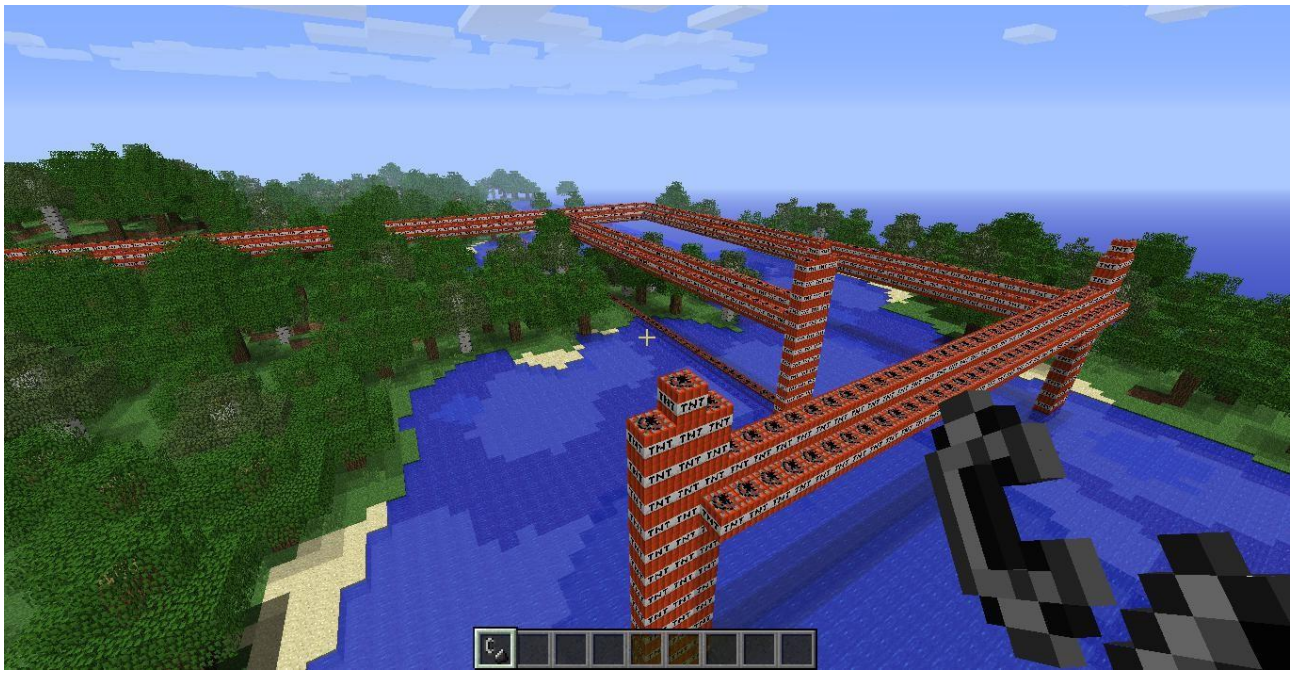
sudo python house.py

Lesson 12

TNT buzzer detector

After we've learned how to generate TNT … why not make a script to detect it?

Imagine you have a metal detector ... if you detect metal it will buzz so you'll know what's beneath you … that's exactly what we are going to make … but with TNT.



Our script will be able to detect TNT up to 15 blocks under us and turn on the buzzer in case there is one! Let's take a look:

```
import RPi.GPIO as GPIO
import time
from mcpi.minecraft import Minecraft

mc = Minecraft.create() # create Minecraft Object

buzzer_pin = 18 # store the GPIO pin number

GPIO.setmode(GPIO.BCM) # change GPIO mode to BOARD
GPIO.setup(buzzer_pin, GPIO.OUT) # setup the pin to OUTPUT

# repeat indefinitely
while True:
    # get player position
    x, y, z = mc.player.getPos()
    # look at every block until block 15
    for i in range(15):
        if mc.getBlock(x, y - i, z) == 46:
            GPIO.output(buzzer_pin, True) # buzz the buzzer on
            time.sleep(0.5) # wait
            GPIO.output(buzzer_pin, False) # turn the buzzer off
            time.sleep(0.5) # wait
```

We run the script forever, get our current position and then for I in range(15) is in order to get up to 15 blocks beneath us, getting the block types and checking if it's equal to block ID 46 which is TNT. If it is, we turn on buzzer to TRUE to make loud buzzing noise and after half a second we turn it back off, easy isn'tit?

You can do the same with other blocks, maybe water? lava? You can also change position to check if it's in front of you or behind you!

In order to download and execute the script, you can run the following commands:

wget https://github.com/Elecrow-RD/CrowPi-for-Raspberry-Pi-5/blob/main/

Minecraft/tnt_detector.py

sudo python tnt_detector.py

Lesson 13

Generating blocks using NFC

Generating NFC blocks is one of the most interesting lessons we'll have. Basically, as we learned, every block in the game has its own unique ID, and we'll use these IDs to write them into the NFC tag and then read them using the CrowPi reader. While reading them - if everything goes by the plan we will generate the ID we read from the tag into the game and combine the virtual life with real life!

This lesson will have 2 parts:

Part I - Generating an NFC tag using python script by selecting block type and ID and writing it into the NFC tag.

Part II - Reading the NFC tags and writing the blocks into the game if we were able to read the NFC block successfully!

Let's spend less time talking and more time acting!

Let’s take a look at the writing script first:

```
import RPi.GPIO as GPIO
from mfrc522 import SimpleMFRC522

reader = SimpleMFRC522()

try:
        text = "64"
        reader.write(text)
        print("Written")
finally:
        GPIO.cleanup()
```

The script is pretty long so we’ll include only the important part which is the actual writing, the rest of the part is user-friendly instructions. During the instructions the script will try to understand what do you actually want to do.

After running the code you will be lead to step-by-step instructions from putting your card to choosing the block so don’t worry if you get confused! The script will guide you.

In order to download and execute the script, you can run the following commands:

wget https://github.com/Elecrow-RD/CrowPi-for-Raspberry-Pi-5/blob/main/Minecraft/nfc

_block_writer.py

sudo python nfc_block_writer.py

Now, after we put our block into the card we want, let's read it by running another script!

Before running the script **make sure your Minecraft is up and running inside your CrowPi, or else the script won't be able to start**!

Again, let's take a look at the most crucial part of the code:

First after recognizing the card on top of the NFC reader our script will try to parse the block type and sub type from inside of the NFC chip.

After parsing out it will search to see if the block is a valid block and exists inside our mini-block-database. If it is we will continue creating a block using the create_block() command we learned in previous lessons.

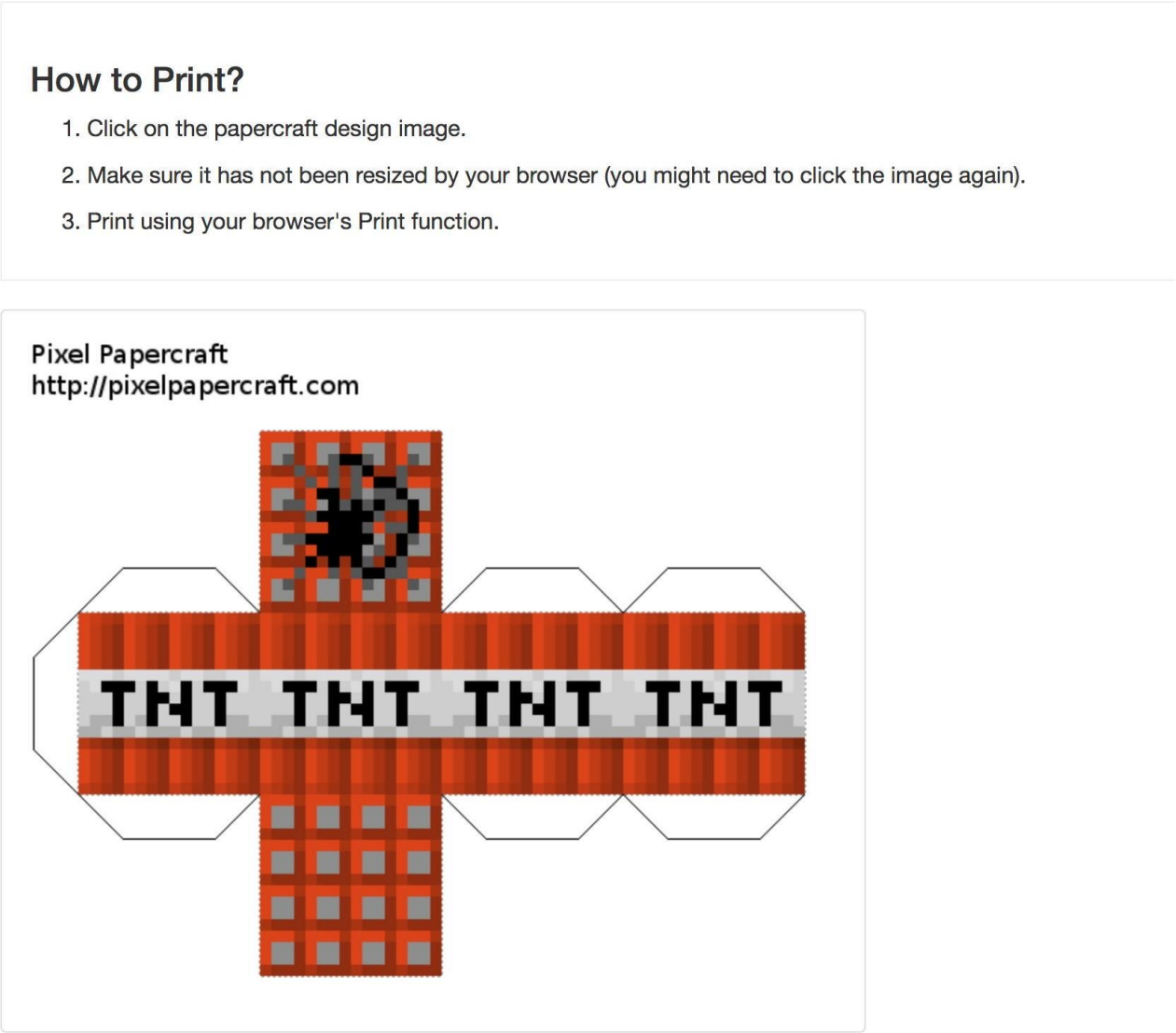
The interesting thing is that if you leave the block on top of the NFC it will keep creating the same block over and over … pretty cool! Of course, with all the lessons you've learned so far - you could easily change that and only create one block each time!

In order to download and execute the script, you can run the following commands:

wget https://github.com/Elecrow-RD/CrowPi-for-Raspberry-Pi-5/blob/main/Minecraft/nfc_block_read.py

sudo python nfc_block_read.py

Make it even better by turning the NFC cards into actual Minecraft blocks!



The website pixelpapercraft.comallows you to download and print Minecraft blocks as paper blocks which were made by the community. We printed those and measured them and they fit just perfectly with the CrowPi NFC cards!

Turn your NFC cards into real Minecraft blocks and program the right block ID into them to combine reality and this virtual game into one!

We can’t wait to see what you’ll do with our CrowPi and Minecraft lessons and what will you make using what you’ve learned during our lessons!