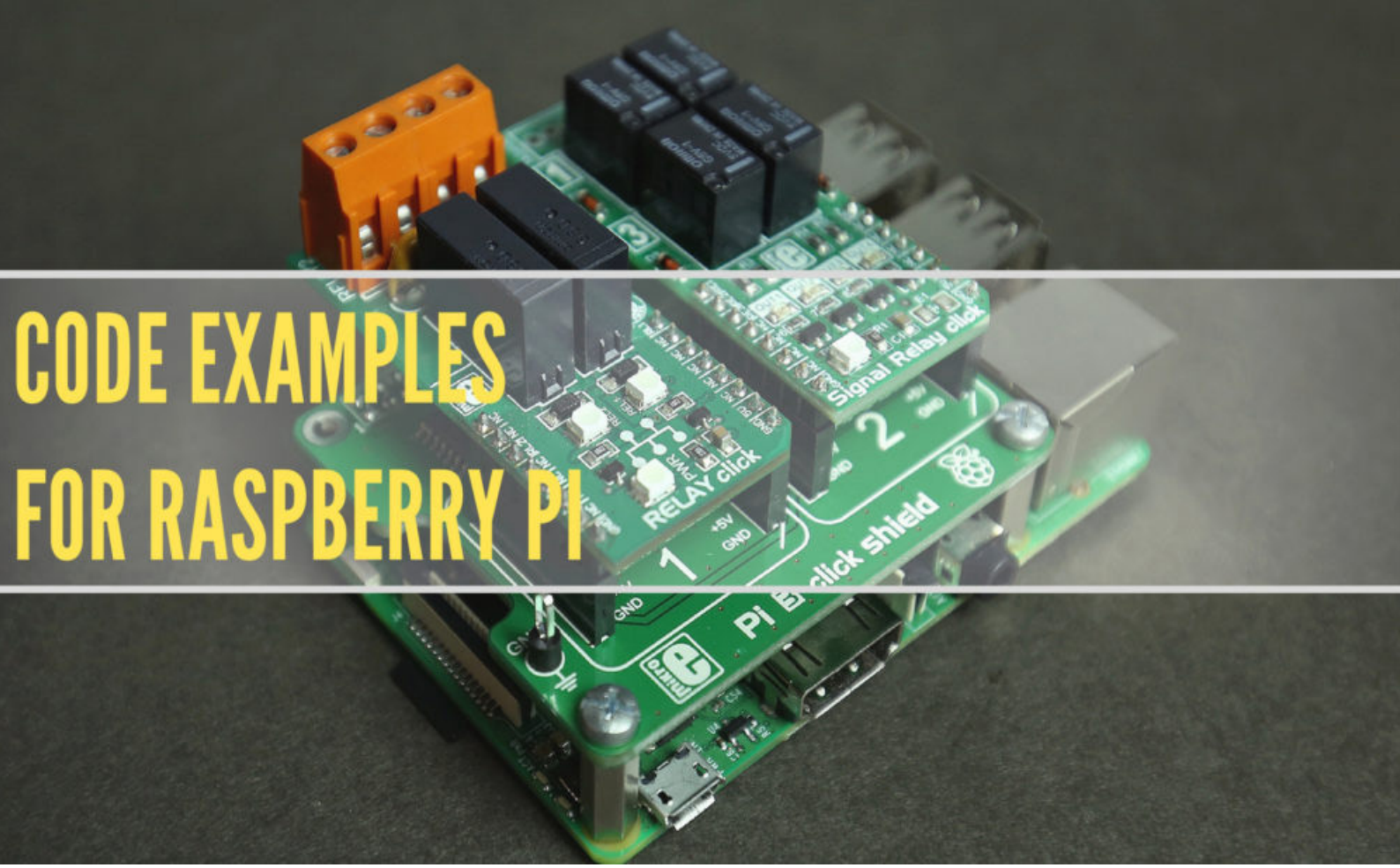


RASPBERRY PI EXAMPLES WITH CLICK BOARDS™

Pi 3 click shield connects the world largest collection of add-on boards – click boards™ with one of the today's most popular embedded platforms – Raspberry Pi.

Here you can find library examples written in Python, a powerful language recommended even for the programming newcomers.

With Pi 3 click shield you are adding two mikroBUS™ sockets to your Raspberry Pi 3, which means that you can experiment with hundreds of click boards™ from the MikroElektronika's range.

The Pi 3 click shield is compatible with Raspberry Pi 3 model B, 2 B, 1 A+ and B+.

System preparation and installation of necessary libraries

Before you begin, you should know how to prepare your Raspberry Pi and how to install the necessary libraries.

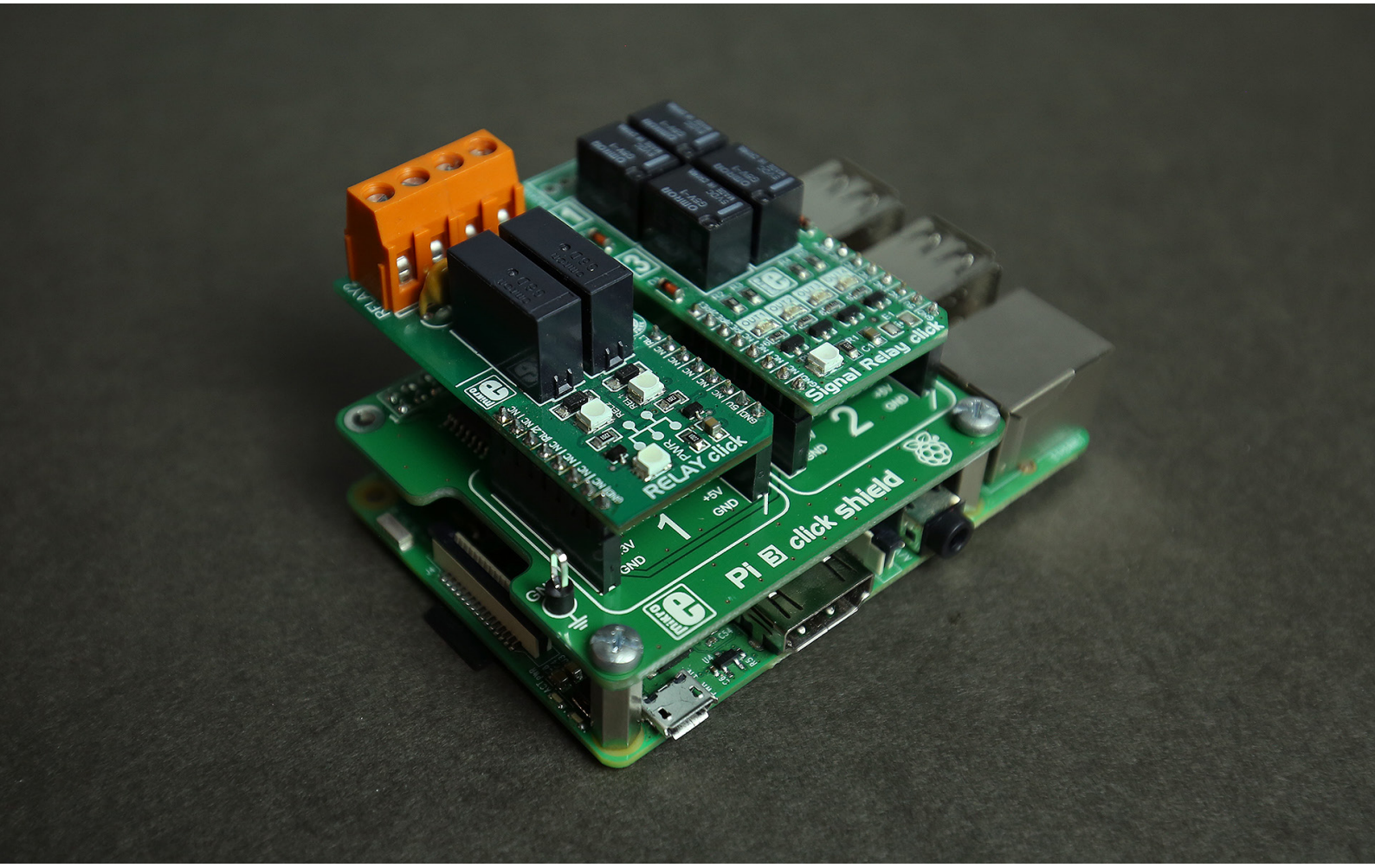
That's why we are giving you step-by-step instructions on how to establish communication with click boards™ on Raspberry Pi via I2C, SPI and UART interfaces.

Python is well known as an effective language that allows programmers to use fewer lines of code than it would be possible in languages such as C or Java. In this article, we are going to show you how easy is the usage of the modules. Let's start with GPIO module.

GPIO

GPIO module usage is pretty easy to understand and to remember.

```python
GPIO.setmode(GPIO.BOARD)
GPIO.setup(5, GPIO.OUT)
GPIO.output(5,GPIO.HIGH)
```



To show an example for GPIO pins used on the Raspberry Pi we've plugged two click boards™, Relay click and Signal Relay click, into the mikroBUS™ sockets which are a part of the Pi 3 click shield.

I2C

I2C tools are necessary when we want to use I2C peripheral on our Raspberry Pi. You can install it by using the following command:

```
pi@Raspberry:~$ sudo apt-get install i2c-tools
```

After that, we must enable loading of i2c-bcm2708 and i2c-dev kernel modules at boot time, by editing the contents of the modules file located in etc sub-folder of your system:

```
pi@Raspberry:~$ sudo nano /etc/modules
```

Next, you need to add the following lines at the end of the file:

```
i2c-bcm2708
i2c-dev
```

Now you can perform the basic test for finding available slave addresses by executing one of the two commands:

`pi@Raspberry:~$ sudo i2cdetect -y 0` or `pi@Raspberry:~$ sudo i2cdetect -y 1`

After the test, you should see the output similar to this one

```
00: -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- 29 -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- 76 -- --
```

Then you need to add the i2c-bcm2708 to the blacklist via the following command:

`pi@Raspberry:~$ sudo nano /etc/modprobe.d/raspi-blacklist.conf` add `i2c-bcm2708`

at the end of the file.

By this time, you would also need a specific Python library which should be installed by the following command:

`pi@Raspberry:~$ sudo apt-get install python-smbus python3-smbus python-dev python3-dev`

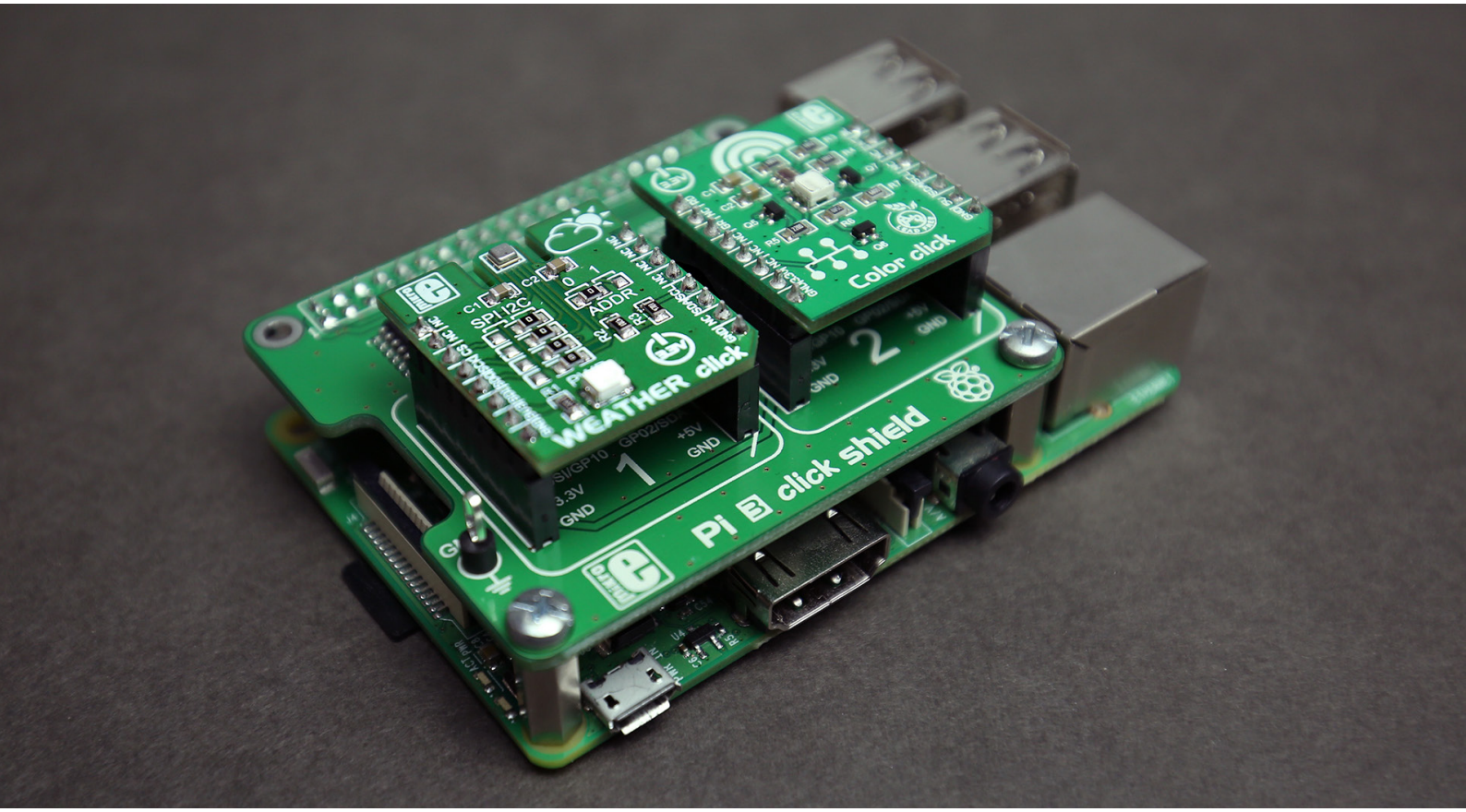
After this, you should be able to use any of I2C peripherals by importing smbus module to your project.

```python

device_address_1 = 0x53
device_address_2 = 0x44

i2c = smbus.SMBus(1)

def write_data( address_reg, data ):
    global device_address_1
    global device_address_2
    id = i2c.read_byte_data(device_address_1, 0x00)
    i2c.write_byte_data(device_address_2, address_reg, data)
```



In order to show an example of the I2C protocol on the Raspberry Pi, we've used Weather click and Color click, again via the Pi 3 click shield.

SPI

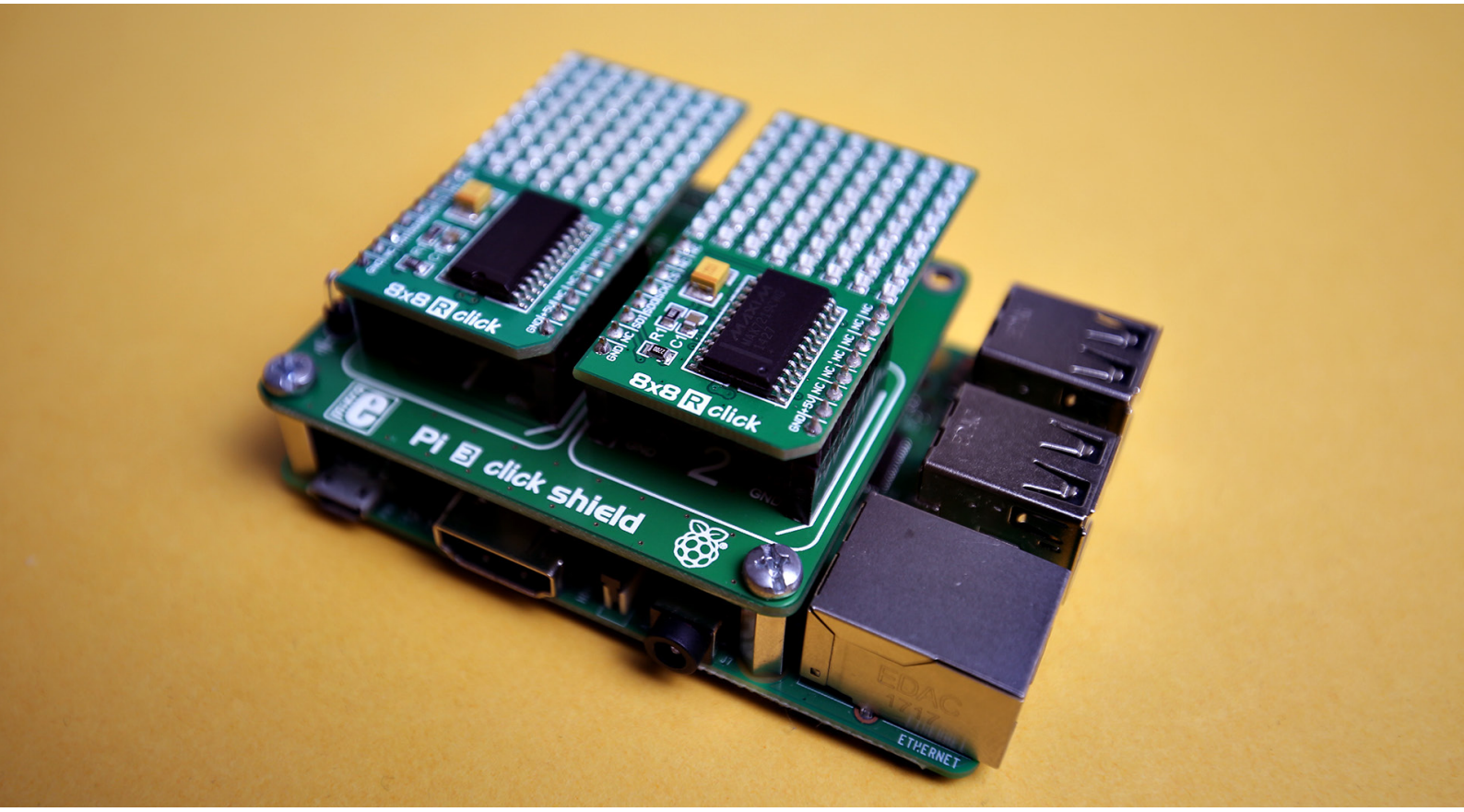
Even easier than I2C, this peripheral requires only an addition of spi-bcm2708 to the blacklist and installation of the specific Python library.

```
pi@Raspberry:~$ sudo apt-get install python3-spidev python-spidev python-dev python3-dev
```

spidev module imports everything needed for the successful SPI communication with the device. Here, unlike with the "bare metal applications", the Linux kernel controls CS or CE pin.

```python
dev1 = spidev.SpiDev()
dev2 = spidev.SpiDev()

def dec_config(cfg1, cfg2):
    dev1.open(0,0)
    dev1.writebytes([cfg1])
    dev2.open(0,1)
    dev2.writebytes([cfg1, cfg2])
```



To demonstrate an example of the SPI protocol used on the Raspberry Pi, we've plugged two 8×8 R click boards™

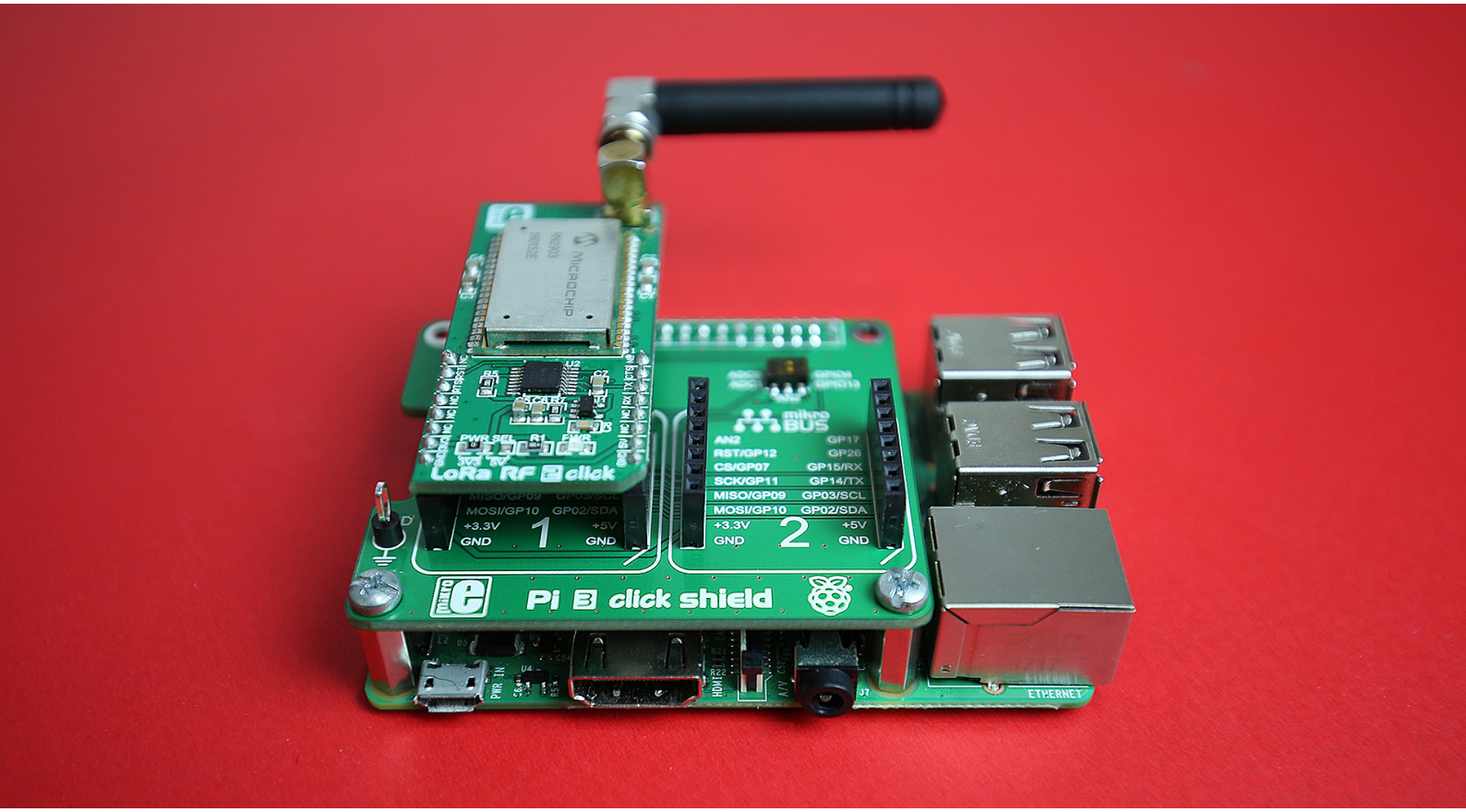
UART

This is the easiest setup. You just need to install pySerial module.

```
pi@Raspberry:~$ python -m pip install pyserial
```

```python
ser = serial.Serial(
                port = '/dev/serial0',
                baudrate = 57600,
                parity=serial.PARITY_NONE,
                stopbits=serial.STOPBITS_ONE,
                bytesize=serial.EIGHTBITS,
                timeout=1
            )

def test():
    ch = ser.read()
    ser.write(ch)
    ser.write('\r\n')
```



Pi 3 click shield and one of the most popular click boards™, LoRa click, helped us in demonstrating the UART protocol used on the Raspberry Pi.

Summary

As you can see, establishing communication with click boards™ on Raspberry Pi via I2C, SPI, and UART interfaces is not a difficult task even if you are a beginner.

We've made examples with some the most popular click boards™ such as LoRa click and 8×8 R click. All examples can be found on LibStock and GitHub.