Python Radio 31: There Goes Your Money!

Using Python to see how contactless transactions work.

Simon Quellen Field

Simon Quellen Field

Follow

24 min read

·

Jan 2, 2025

Listen

Share

More

Press enter or click to view image in full size

Money flying away after a cellphone transaction.

MidJourney

Radio Frequency Identification (RFID) is all around you. Your passport has a chip in it that works without any batteries to communicate with customs and security at airports. Credit cards have them. Smartphones have them.

Android phones have Near Field Communication chips that can read and write to these devices.

When you are paying at a grocery store, you want to be sure you aren’t accidentally paying for someone else’s purchase, just because the register picked up a signal from your phone or card.

NFC relies on Near Field Communication, which works only from a couple of centimetres away. It won’t read your card unless you almost press it onto the reader.

How can it do this without a battery?

The reader emits a signal at 13.56 megahertz into a coil of wire. If a similar coil of wire is close enough, energy from the one coil will couple to the other.

The second coil is attached to a tiny chip that gets power when it is close to the first coil.

However the chip has no way to store enough power to transmit a signal. Instead, it uses a neat trick. It periodically puts a load on the power, in a bit stream that carries data.

The sending coil sees this drain on its power and reads that signal to identify the card and see what information is stored there. It’s a bit like sending data to someone giving you water in a hose by opening and closing the nozzle in Morse code. They can feel the pressure change at their end.

Writing is easier. The device with power just sends a signal to the chip, which then stores the information in non-volatile memory. This is typically good for over a million writes.

NFC cards are used to access locked doors at places like gyms or hotel rooms. Identity information and access permission codes are written into the cards at the front desk and the user now has free access to the facility.

There are a number of cards and tags that you can buy. As you might expect, these have to be cheap and are often sold in bulk. So I have a bunch of them:

Press enter or click to view image in full size

A selection of NFC tags and cards.

Photo by author.

My favourites are the little transparent ones, but I had trouble writing to them (except when using my smartphone). In this article, we will be using the credit card size.

For the reader, I chose this nice little board from Adafruit — the RFID-RC522.

Press enter or click to view image in full size

The RFID-RC522 module from Adafruit.

Photo by author.

We will talk to it using an ESP32 we have not used before, the Lolin S2 Mini.

I started this project using the Wemos D1 Mini (an ESP8266 board), but the driver required more RAM than it had. But the S2 Mini has a similar form factor and many advantages. It is an ESP32, so much bigger and faster. It has a USB-C port, which is much easier to deal with than the old Micro-USB.

It has twice as many I/O pins. Which is great, but to fit them in, they had to make two rows, so it is not breadboard-friendly anymore.

It comes with rows of header pins you can solder on, but since that won’t help breadboarding, it seems of only marginal use. Instead, I brought out my own set of socket headers (shown at the upper right). Now I can just plug in Dupont rainbow ribbon cable connectors just like on a breadboard.

Press enter or click to view image in full size

The Lolin S2 Mini ESP32 computer board.

Photo by author.

The result turned out like this:

Press enter or click to view image in full size

The RFID-RC522 connected to the Lolin S2 Mini.

Photo by author.

ESP Pin “3V3” (red) connected to 522 Pin “3.3v”

ESP Pin “17” ((green) connected to 522 Pin “MISO”

ESP Pin “11” (white) connected to 522 Pin “IRQ”

ESP pin “14” (orange) connected to 522 Pin “SCK”

ESP Pin “13” (yellow) connected to 522 Pin “MOSI”

ESP Pin “15” (blue) connected to 522 Pin “SDA”

ESP Pin “GND” (black) connected to 522 Pin “GND”

ESP Pin “17” (brown) connected to 522 Pin “RST”

The S2 Mini has 2 whole megabytes of RAM. That sounds laughable to those of us used to demanding tens of gigabytes just to surf the web, but it is a real luxury after spending weeks trying to cram a big driver into the ESP8266.

But that means you have to choose the version of Micropython that supports external RAM. See this page for the download. Hold the 0 button while powering up to get it into programming mode. The rest of the instructions for loading Micropython can be found earlier in this series, or at micropython.org.

Before launching into code, let’s look at the program running for a bit.

Hit the ENTER key when the program starts, and you get:

R for Read, F for Format, E for Erase, W for Write:

Place the card on the reader and type r and ENTER.

Please place card on reader

Card 0XDA8A0EC3 [0xC3, 0x0E, 0x8A, 0xDA]

READ

00 S00 B0: Len 16 C3 0E 8A DA 9D 08 04 00 62 63 64 65 66 67 68 69 ........bcdefghi

01 S00 B1: Len 16 7E 01 03 E1 03 E1 03 E1 03 E1 03 E1 03 E1 03 E1 ~...............

02 S00 B2: Len 16 7E 01 03 E1 03 E1 03 E1 03 E1 03 E1 03 E1 03 E1 ~...............

03 S00 B3: Len 16 00 00 00 00 00 00 78 77 88 FF 00 00 00 00 00 00 ......xw........

04 S01 B0: Len 16 68 74 74 70 73 3A 2F 2F 73 63 69 74 6F 79 73 2E <https://scitoys>.

05 S01 B1: Len 16 63 6F 6D 00 00 00 00 00 00 00 00 00 00 00 00 00 com.............

06 S01 B2: Len 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

07 S01 B3: Len 16 00 00 00 00 00 00 7F 07 88 FF 00 00 00 00 00 00 ................

08 S02 B0: Len 16 68 74 74 70 73 3A 2F 2F 70 75 72 65 66 69 78 69 <https://purefixi>

09 S02 B1: Len 16 6F 6E 2E 63 6F 6D 00 00 00 00 00 00 00 00 00 00 on.com..........

10 S02 B2: Len 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

11 S02 B3: Len 16 00 00 00 00 00 00 7F 07 88 FF 00 00 00 00 00 00 ................

12 S03 B0: Len 16 68 74 74 70 73 3A 2F 2F 6E 65 74 72 6F 67 6C 79 <https://netrogly>

13 S03 B1: Len 16 63 65 72 69 6E 65 2E 63 6F 6D 00 00 00 00 00 00 cerine.com......

14 S03 B2: Len 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

15 S03 B3: Len 16 00 00 00 00 00 00 7F 07 88 FF 00 00 00 00 00 00 ................

16 S04 B0: Len 16 68 74 74 70 73 3A 2F 2F 62 69 72 64 66 61 72 6D <https://birdfarm>

17 S04 B1: Len 16 2E 6F 72 67 00 00 00 00 00 00 00 00 00 00 00 00 .org............

18 S04 B2: Len 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

19 S04 B3: Len 16 00 00 00 00 00 00 7F 07 88 FF 00 00 00 00 00 00 ................

20 S05 B0: Len 16 54 72 75 74 68 20 69 73 20 61 20 73 68 61 64 6F Truth.is.a.shado

21 S05 B1: Len 16 77 2C 20 68 69 73 20 70 61 73 73 69 6F 6E 20 61 w,.his.passion.a

22 S05 B2: Len 16 20 6B 69 73 73 2C 20 77 68 65 6E 20 6B 6E 6F 77 .kiss,.when.know

23 S05 B3: Len 16 00 00 00 00 00 00 7F 07 88 FF 00 00 00 00 00 00 ................

24 S06 B0: Len 16 6C 65 64 67 65 20 69 73 20 70 6F 77 65 72 20 61 ledge.is.power.a

25 S06 B1: Len 16 6E 64 20 69 67 6E 6F 72 61 6E 63 65 20 62 6C 69 nd.ignorance.bli

26 S06 B2: Len 16 73 73 2E 00 00 00 00 00 00 00 00 00 00 00 00 00 ss..............

27 S06 B3: Len 16 00 00 00 00 00 00 7F 07 88 FF 00 00 00 00 00 00 ................

28 S07 B0: Len 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

29 S07 B1: Len 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

30 S07 B2: Len 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

31 S07 B3: Len 16 00 00 00 00 00 00 7F 07 88 FF 00 00 00 00 00 00 ................

32 S08 B0: Len 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

33 S08 B1: Len 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

34 S08 B2: Len 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

35 S08 B3: Len 16 00 00 00 00 00 00 7F 07 88 FF 00 00 00 00 00 00 ................

36 S09 B0: Len 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

37 S09 B1: Len 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

38 S09 B2: Len 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

39 S09 B3: Len 16 00 00 00 00 00 00 7F 07 88 FF 00 00 00 00 00 00 ................

40 S10 B0: Len 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

41 S10 B1: Len 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

42 S10 B2: Len 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

43 S10 B3: Len 16 00 00 00 00 00 00 7F 07 88 FF 00 00 00 00 00 00 ................

44 S11 B0: Len 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

45 S11 B1: Len 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

46 S11 B2: Len 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

47 S11 B3: Len 16 00 00 00 00 00 00 7F 07 88 FF 00 00 00 00 00 00 ................

48 S12 B0: Len 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

49 S12 B1: Len 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

50 S12 B2: Len 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

51 S12 B3: Len 16 00 00 00 00 00 00 7F 07 88 FF 00 00 00 00 00 00 ................

52 S13 B0: Len 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

53 S13 B1: Len 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

54 S13 B2: Len 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

55 S13 B3: Len 16 00 00 00 00 00 00 7F 07 88 FF 00 00 00 00 00 00 ................

56 S14 B0: Len 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

57 S14 B1: Len 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

58 S14 B2: Len 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

59 S14 B3: Len 16 00 00 00 00 00 00 7F 07 88 FF 00 00 00 00 00 00 ................

60 S15 B0: Len 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

61 S15 B1: Len 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

62 S15 B2: Len 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

63 S15 B3: Len 16 00 00 00 00 00 00 7F 07 88 FF 00 00 00 00 00 00 ................

You get a dump of all the data on the card. If a character is printable, it shows up on the right, otherwise, there’s just a period. Send me a note if you see the poem.

Next, we’ll try formatting the card, then erasing the card, and finally writing five different records of varying lengths:

R for Read, F for Format, E for Erase, W for Write:

F

Please place card on reader

Card 0XDA8A0EC3 [0xC3, 0x0E, 0x8A, 0xDA]

FORMAT

Writing MAD sector failed.. Try erasing the card

Can’t format

E

Please place card on reader

Card 0XDA8A0EC3 [0xC3, 0x0E, 0x8A, 0xDA]

ERASE

Erased

F

Please place card on reader

Card 0XDA8A0EC3 [0xC3, 0x0E, 0x8A, 0xDA]

FORMAT

................Done

W

Please place card on reader

Card 0XDA8A0EC3 [0xC3, 0x0E, 0x8A, 0xDA]

WRITE

Please place card on reader

Card 0XDA8A0EC3 [0xC3, 0x0E, 0x8A, 0xDA]

WRITE

Please place card on reader

Card 0XDA8A0EC3 [0xC3, 0x0E, 0x8A, 0xDA]

WRITE

Please place card on reader

Card 0XDA8A0EC3 [0xC3, 0x0E, 0x8A, 0xDA]

WRITE

Please place card on reader

Card 0XDA8A0EC3 [0xC3, 0x0E, 0x8A, 0xDA]

WRITE

Note that the card has access permissions that prevent us from formatting at first. We have to erase all the data before we can write to certain places (and reformatting wants to write to a place the original format had protected).

We’ll see a little bit about that when we look at our main.py file.

That big driver I talked about is in a file called mfrc522.py. We’ll just look at how the class it defines is constructed before diving way in. The instance of that class is constructed in main.py. The class itself is MFRC522:

Class Ndef:

READ = const(0)

ERASE = const(1)

WRITE = const(2)

FORMAT = const(3)

ISNTAG = const(4)

DUMPNTAG = const(5)

MAX\_BLOCKS = const(64)

Def \_\_init\_\_(self):

From mfrc522 import MFRC522

Self.reader = MFRC522(sck = 14, mosi = 13, miso = 12, rst = 17, cs = 15, spi\_id = 1)

Def checksum(self, data):

Crc = 0xc7

For byte in data:

Crc ^= byte

For \_ in range(8):

Msb = crc & 0x80

Crc = (crc << 1) & 0xff

If msb:

Crc ^= 0x1d

Return crc

Def access\_setup(self):

From RfidAccess import RfidAccess

Return RfidAccess()

Def Process(self, cmd, data = None, sect = None, blk = None):

From utime import sleep\_ms

defaultKey = [0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF]

firstSectorKey = [0xA0, 0xA1, 0xA2, 0xA3, 0xA4, 0xA5]

nextSectorKey = [0xD3, 0xF7, 0xD3, 0xF7, 0xD3, 0xF7]

print(“Please place card on reader”)

for i in range(50):

self.reader.init()

access = self.access\_setup()

(stat, tag\_type) = self.reader.request(self.reader.REQIDL)

If stat == self.reader.OK:

(stat, uid) = self.reader.SelectTagSN()

If stat == self.reader.OK:

Uid\_str = hex(int.from\_bytes(bytes(uid), “little”, False)).upper()

Uid\_hex = self.reader.tohexstring(uid)

Print(“Card {} {}”.format(uid\_str, uid\_hex))

If cmd == READ:

Print(“READ”)

#read MAD sector (first sector)

If self.reader.MFRC522\_DumpClassic1K(uid, Start = 0, End = 4, keyA = firstSectorKey) == self.reader.OK:

#read the rest of the card

Self.reader.MFRC522\_DumpClassic1K(uid, Start = 4, End = 64, keyA = nextSectorKey)

Else:

If self.reader.MFRC522\_DumpClassic1K(uid, Start = 0, End = 4, keyA = defaultKey) == self.reader.OK:

#read the rest of the card

Self.reader.MFRC522\_DumpClassic1K(uid, Start = 4, End = 64, keyA = defaultKey)

Return True

Elif cmd == WRITE:

Print(“WRITE”)

While len(data):

Buf = bytearray(data[0:16], “utf-8”)

While len(buf) < 16:

Buf.append(0)

If self.reader.writeSectorBlock(uid, sect, blk, buf, keyB = defaultKey) == self.reader.ERR:

Print(“Write failed sector”, sect, “, block”, blk)

Data = data[16:]

Blk += 1

If blk == 3:

Sect += 1

Blk = 0

Return True

Elif cmd == ERASE:

Print(“ERASE”)

# set default access

Access.decodeAccess(0xff, 0x07, 0x80)

Block3 = access.fillBlock3(keyA = defaultKey, keyB = defaultKey)

# print(“Reset Mad Sector (first sector)”)

#reset first sector

Self.reader.writeSectorBlock(uid, 0, 3, block3, keyB = defaultKey)

#erase block1 and 2

Datablock = 16 \* [0]

Self.reader.writeSectorBlock(uid, 0, 1, datablock, keyB = defaultKey)

Self.reader.writeSectorBlock(uid, 0, 2, datablock, keyB = defaultKey)

#reset all other sectors

For s in range(1, 16):

# permission to default

# print(“Reset sector “, s)

Self.reader.writeSectorBlock(uid, s, 3, block3, keyB = defaultKey)

For b in range(3):

# put all data to zero block 0, 1 and 2

Self.reader.writeSectorBlock(uid, s, b, datablock, keyB = defaultKey)

Print(“Erased”)

Return True

Elif cmd == FORMAT:

Print(“FORMAT”)

#set MAD sector

# first fill block permission

Access.setTrailerAccess(keyA\_Write = access.KEYB, access\_Read = access.KEYAB, access\_Write = access.KEYB,

keyB\_Read = access.NEVER, keyB\_Write = access.KEYB)

access.setBlockAccess(access.ALLBLOCK, access\_Read = access.KEYAB, access\_Write = access.KEYB,

access\_Inc = access.NEVER, access\_Dec = access.NEVER)

block3 = access.fillBlock3(keyA = firstSectorKey, keyB = defaultKey)

# print(“Len of block3 is”, len(block3))

#Write the sector access

If self.reader.writeSectorBlock(uid, 0, 3, block3, keyA = defaultKey) == self.reader.ERR:

If self.reader.writeSectorBlock(uid, 0, 3, block3, keyA = firstSectorKey) == self.reader.ERR:

Print(“Writing MAD sector failed.. Try erasing the card”)

Return False

Else:

Print(“.”, end = “”)

B1 = [0x14, 0x01, 0x03, 0xE1, 0x03, 0xE1, 0x03, 0xE1, 0x03, 0xE1, 0x03, 0xE1, 0x03, 0xE1, 0x03, 0xE1]

B1[0] = self.checksum(b1[1:]) # I know this is already ok but just to demonstrate the CRC

Self.reader.writeSectorBlock(uid, 0, 1, b1, keyB = defaultKey)

# b2 = [0x03, 0xE1, 0x03, 0xE1, 0x03, 0xE1, 0x03, 0xE1, 0x03, 0xE1, 0x03, 0xE1, 0x03, 0xE1, 0x03, 0xE1]

Self.reader.writeSectorBlock(uid, 0, 2, b1, keyB = defaultKey)

#set permission for all other sectors

Access.setTrailerAccess(keyA\_Write = access.KEYB, access\_Read = access.KEYAB, access\_Write = access.KEYB,

keyB\_Read = access.NEVER, keyB\_Write = access.KEYB)

access.setBlockAccess(access.ALLBLOCK, access\_Read = access.KEYAB, access\_Write = access.KEYAB,

access\_Inc = access.KEYAB, access\_Dec = access.KEYAB)

block3 = access.fillBlock3(keyA = nextSectorKey, keyB = defaultKey)

#Write all next sectors access

For sector in range(1, 16):

If self.reader.writeSectorBlock(uid, sector, 3, block3, keyA = defaultKey) == self.reader.ERR:

Print(“\nWriting to sector “, sector, “ Failed!”)

Return False

Else:

Print(“.”, end = “”)

#force sector 1 to be 1 record empty

Block = 16 \* [0]

Block[2] = 0xfe

If self.reader.writeSectorBlock(uid, 1, 0, block, keyB = defaultKey) == self.reader.ERR:

Print(“Unable to set first NDEF record!”)

Return False

Elif cmd == ISNTAG:

Print(“Is NTAG” if self.reader.IsNTAG() else “Not NTAG”)

Elif cmd == DUMPNTAG: