

# A Multiple Badge Architectures Open Source RFID Reader with Insight Regarding Room Occupants

Jordan Barrett<sup>1</sup>, Adnan Shaout<sup>2\*</sup> 

<sup>1, 2</sup>Electrical and Computer Engineering Department, College of Engineering and Computer Science, The University of Michigan - Dearborn, Dearborn, Michigan, USA  
E-mail: shaout@umich.edu

Received: April 20, 2022

Revised: July 17, 2022

Accepted: July 22, 2022

**Abstract** - The preferred architecture and the source company for radio frequency identification (RFID) badge access cards can vary over time within the same company, depending on what management may envision for the future of working. Additionally, the preferred badge architecture and the company that makes those badges can vary from company to company, and from original equipment manufacturer (OEM) to their supplier partners. These circumstances create some inefficiencies amongst the workforce. Within the same engineering campus of a large OEM, many employees may still have the old key ring style badge, but some of the new RFID device readers may no longer accept this outdated badge architecture, which means these employees may not have access to certain areas until they get their new smart badge ID card. This makes doing their job much more difficult, potentially forcing them to find new ways to access the room or area that they need. Additionally, due to differences in badge architecture, an OEM's supplier partners do not have access to areas of the OEM's testing site or offices, even ones in which the OEM would prefer that they have permission to access in order to increase work efficiency (including test tracks and special laboratories). This paper presents a new open sourced RFID card reader which is designed, tested and implemented to read both of the most popular badge architectures (key ring badges and smart cards). The new proposed system also includes a unique function which shows the employee more information about the occupants inside of the room that they are trying to access. After reflecting on the state-of-the-art, the main selling point of the proposed system is that it recognizes multiple different badge architectures, and it doesn't require the end user to source their RFID devices from a certain company. It provides other benefits including allowing suppliers and OEMs to seamlessly share collaborative spaces, and ensuring older versions of badges don't become obsolete.

**Keywords** - RFID; Badge reader; Embedded system; Real time system.

## 1. INTRODUCTION

Designing radio frequency identification (RFID) reader embedded system which is able to identify users who have both a smart card chip badge and a key ring badge is needed by many companies [1, 2]. By identifying two separate badge architectures, we can ensure that employees with the 'old' architecture (key ring badges) still have access to the same rooms that the new smart card chip badges allow. In many companies, such as Ford - various suppliers who need to have access to specific rooms and test labs - have key ring badges, whereas Ford has smart card badges. The badge readers at Ford do not work for the supplier's key ring badge. It would be nice to have a reader that accepts both architectures, to increase work efficiency and ease cross-company collaboration. Additionally, some test tracks still require separate badge architecture to access them; meaning test track engineering personnel are required to carry around two badges. Hence, a badge reader that accepts both types of badges would be very useful.

In the current state-of-the-art, there are various badge readers in the market. Crown Security Products LLC has a badge reader which only accepts their own smart chip badges [3]. This product appears to primarily be aimed at preventing what they call 'buddy-

\* Corresponding author

punching,' where a co-worker can punch in a worker that is not actually on-site working. This essentially allows the workers to be paid while they are not working. Although this seems like a useful concept, this is not what our proposed system intends to accomplish. In fact, this is a common theme when researching the current state-of-the-art: for security purposes, most companies seem to demand the usage of their specific RFID device with their RFID reader.

Alphacard also has an RFID solution and does allow the user to have both a smart card chip badge and a key ring [4]. However, in both cases, you need to buy Alphacard's specific RFID devices. It would be better to have a more open sourced approach; to address the specific use case that a supplier would want to access an original equipment manufacturer (OEM) shared collaboration space. Suppliers and OEMs are not likely to have sourced RFID devices from the same company.

One of the most relevant articles that details a 'context-aware' system using RFID is a paper by Corches et al. [5]. Essentially, the authors were seeking to gather an RFID user's location information and combine it with environmental data (humidity, temperature, air quality, etc.) to make certain welcomed changes to the user's environment. For example, one item they were trying to implement was changing the heat and A/C levels based on this automated feedback from RFID.

After reflecting on the state-of-the-art, the main selling point of the proposed system in this paper is that it recognizes multiple different badge architectures, and it doesn't require the end user to source their RFID devices from a certain company. The proposed system fits in the market a bit differently than the system in [5]. Instead of preventing 'buddy punching,' or making automatic changes to the user's environment, our proposed system aims to provide other benefits, including allowing suppliers and OEMs to seamlessly share collaborative spaces, and ensuring older versions of badges not to become obsolete. The target customer for our product is large corporations, including OEMs and suppliers. The proposed system is feasible due to the fact that most of the RFID pieces are not new technology, and the required materials seem to be relatively cheap (under \$200 in total) and readily available despite the ongoing chip shortage.

An embedded system which reads an RFID chip within a key ring or smart card badge will be presented in this paper. The proposed system should work as follows: when the user swipes an RFID device, independent of the RFID architecture (key ring or smart badge), the badge ID is displayed on the monitor, along with a welcome message. Also, added additional functionality to the proposed design which enables it to identify the current occupants in the room which the user is swiping into, as well as associate a name with the badge ID, creating a friendlier and warm experience for the user. The added functionality not only enables the reader - when a person swipes their RFID device - to display the normal welcome message with the badge ID, but it also will display the user's name, and then it will tell the user who is in the room that they are trying to enter. Lastly, if the user has already swiped into the room and is re-entering, the system will recognize that as well, and tell him that he already has access and to please enter.

The rest of the paper is organized as follows: section 2 will present the proposed system and its design and implementation. Section 3 will present the test of the proposed system and section 4 will present the conclusions.

## 2. THE PROPOSED SYSTEM

In order to stand out from the crowd and satisfy the market needs, it has been decided that the proposed system will have the following functional requirements so that it meets and exceeds the-state-of-the-art:

- a) Identify the user within 3 seconds of swiping the badge, since employees do not want to swipe their badge and then wait about 30 s, or one minute, or more for the device to recognize their swipe
- b) Correctly identify 80% or greater of people swiping their badges; employees and managers will be very upset if their brand new badge reader cannot correctly identify a badge ID. 80% is the absolute minimum requirement
- c) Must be able to identify both Smart Card Chip badges as well as key ring badges. The badge reader must identify both badge architectures, because this is a key piece of the product concept and this is what makes the product stand out amongst the state-of-the-art
- d) Must be compatible with 13.56MHz Card Reader Frequency, since the 13.56 MHz frequency will ensure that this device doesn't interfere with AM/FM radio signals [6]
- e) Must be able to identify cards swiped within 0.25" of the reader but not cards swiped greater than 5" from the reader. To ensure that the device reads only the cards that have been deliberately swiped onto the reader, we must specify a 'reader range.' Otherwise, the device could inadvertently read RFID chips of people walking by the reader on their way to somewhere else
- f) Operates safely in ambient temperatures of 5-27° C. This is the operating temperature of office buildings, accounting for extreme cases when the A/C or heat may not be working
- g) Ensure that the functional operating temperature of the final system is below 80° C. The system itself cannot get too hot, or else it will burn employees that walk by and touch it, or swipe their badge on it and make contact with it. This requirement applies to the microprocessor temperature, not the surface temperature of the reader, although these two numbers are related.

We also have decided to use the following for implementing the proposed system:

- a) Must use Python (can also use other languages, but must use Python at some point).
- b) Utilize Raspberry Pi.
- c) Utilize Raspbian operating system (OS).

These items have been chosen because Python is an important real time language, Raspberry Pi is a widely popular microprocessor (so it will be easy to find), and the Raspbian OS is compatible with most Raspberry Pi add on devices and it is more than powerful and fast enough for the proposed system application.

The proposed system will be tested - once it is fully implemented - using the following steps:

- a) Amend badge reader script to utilize Python's 'time.clock()' function to accurately measure the time between the card being swiped and the display to the monitor.
- b) Swipe five badges (from both architectures) and read the screen to ensure that at least four badges were read correctly. To confirm the cards have been read, look for the badge ID on the screen.

- c) Swipe a smart card chip badge and check for the badge ID on the screen. Swipe a key ring badge and check for the badge ID on the screen.
- d) Buy an RFID chip and RFID devices that support 13.56 MHz card reader frequency.
- e) Swipe a card within 0.25" from the reader and verify that the message ID appears on the screen.
- f) Swipe a card greater than 5" from the reader and verify that the message ID does not appear on the screen.
- g) Cool the environment down to 5° C and check the temperature of the microprocessor by entering the command 'vcgencmd measure\_temp' into Linux. Once the temp reaches 5° C, scan a card over the reader and look for the ID on the screen. Repeat by heating the system to 27° C using a space heater and scan a card over the reader and look for the ID on the screen.
- h) Run the system repeatedly in an ambient environmental temperature of 27° C. Run the system for 20 min, scanning several cards throughout the 20 min. Scan 10 cards in the last minute (the goal is to heat the system up). Check the temperature of the microprocessor by entering the command 'vcgencmd measure\_temp' into Linux.

## 2.1. The Proposed System Design

Fig. 1 shows the design for the Software of the proposed system (both the original in black and the extra features in green).

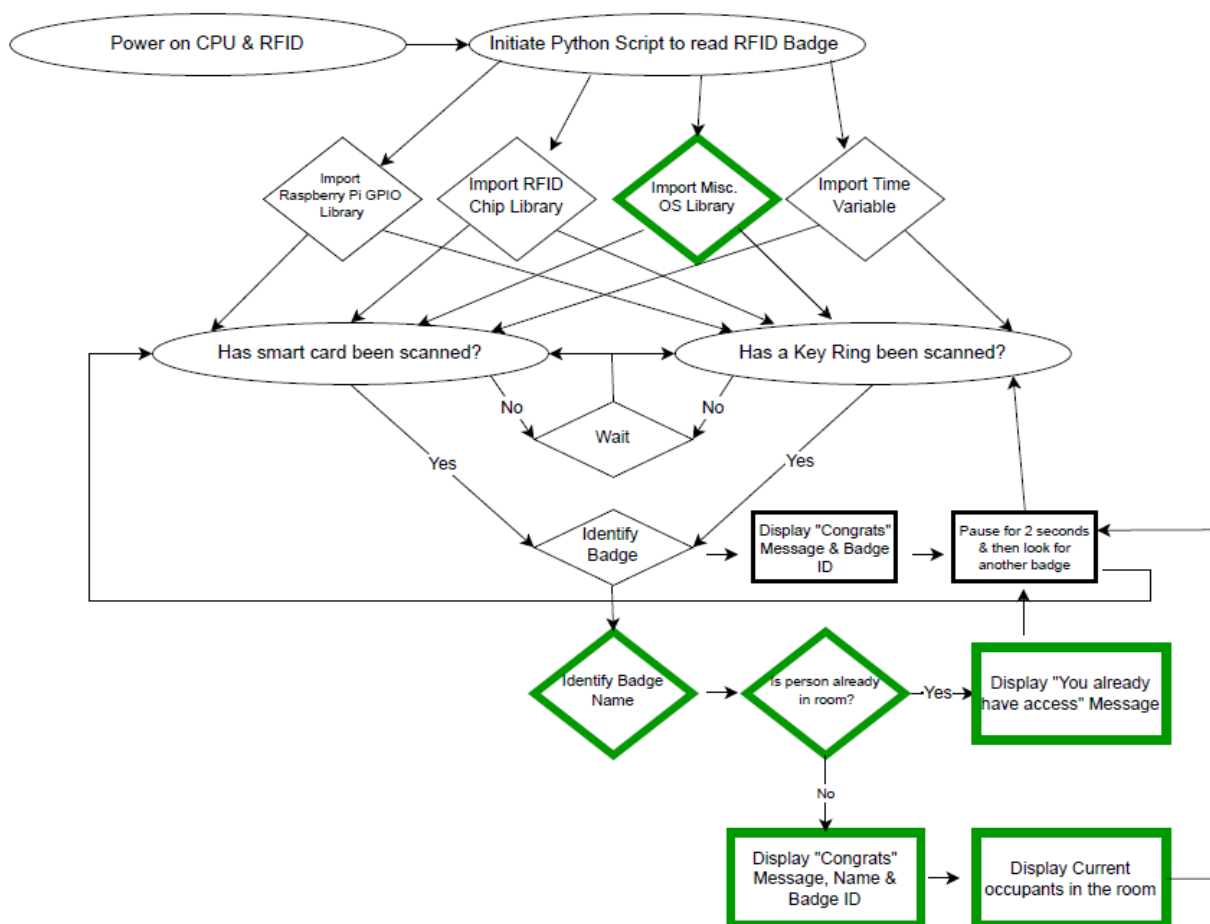


Fig. 1. The flowchart for the software system.

According to the flowchart, the user must first power on the system and initiate a Python script. The script will then import the necessary libraries needed for the various operations and function calls to read and write various RFID tags. It will have the ability to read both a key ring badge and a smart card badge. After that, it will identify the badge ID and display a welcome message to the user. A key piece here is that the system will pause for 2 s before looking for another badge. This is to prevent the situation where the card reader is constantly reading the RFID tag while the tag is in range of the reader. Even if the user swipes quickly over the reader, the system can sometimes pick that card up more than once, so it is necessary to address this issue by implementing a pause.

Additional functionality was added as described in the green boxes above in Fig. 1. The design which enables the system to display a unique badge name for each badge as well as display who is currently in the room that the user is trying to access is shown in Fig. 2. Also, if a user has already swiped into the room and he is returning, the system will identify that he has already access to the room and tell him to proceed. It will not add his name twice to the 'room occupants' list. This additional functionality is considered extra work, and was achieved through additional software only, which we found to be very cost effective and efficient.

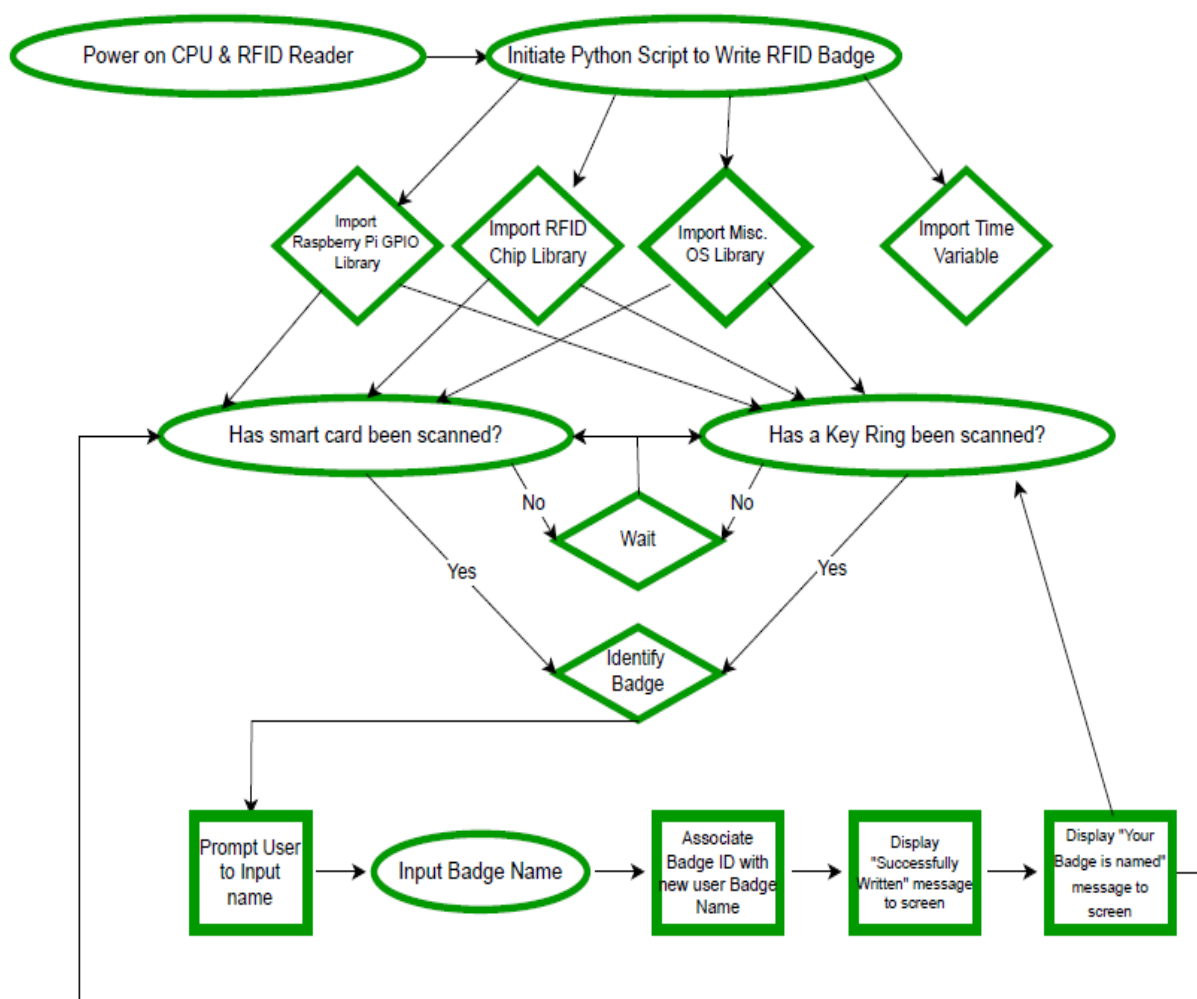


Fig. 2. Software design flowchart for an algorithm which writes a custom name to a badge ID.

Additionally, to enable the extra functionality detailed in the green boxes above, we needed to create another software design flowchart for a function which would write a custom name to a badge ID. This design flowchart is shown in Fig. 2. To describe this design flowchart a bit more; the user will power on the system, and then the software will have to input the necessary libraries and then identify the badge ID of a swiped RFID device (regardless of the device architecture). After that, the script will prompt the user to input the unique badge name, and then the software will associate that badge name to the badge ID and display a 'success' message to the user. The above software design is crucial to the extra functionality that we have added. Without the ability to assign unique names to badge IDs, the system would be unable to identify the users (by name) who are already in the room that a person is trying to gain access to.

Lastly, a few additional software design considerations had to be made. We chose an RFID reader chip and a microprocessor which both support SPI communication, so we had to enable serial peripheral interface (SPI) communication on our Raspberry Pi. This requires downloading the SPI Development Libraries from the Raspberry Pi website [7]. We also needed the MFRC522 library which goes along with our RFID reader chip, and enables SPI communication between the reader and the Raspberry Pi. This library is available on GitHub [8]. We had to download and activate the Raspbian OS on our Pi, which comes separately on an SD card [9]. To make the script which will run on our microprocessor and integrates the RFID reader, we used the latest Python version (3.7.3), which came pre-installed on our Raspberry Pi.

Since we did an integrated hardware (HW) and software (SW) system, it is necessary to justify our HW design, in addition to our SW design. We decided to use the Philips MFRC522 card reader chip because it meets our aforesaid design requirements. The chip used was purchased from Amazon, along with several other chips to use as a backup, in case this one didn't work [6].

A Raspberry Pi 4 was used as for CPU in the proposed system, because it met our design requirements, namely:

- It operates correctly in the temperature range of 5-27° C
- It is fast enough to identify the badge ID within 3 seconds
- The CPU can operate correctly below 80° C

We chose the lowest level Raspberry Pi model, which comes with 2 GB of RAM and a 1.5 GHz ARM Cortex-A72 CPU, which has more capability than our needs.

## 2.2. The Proposed System Implementation

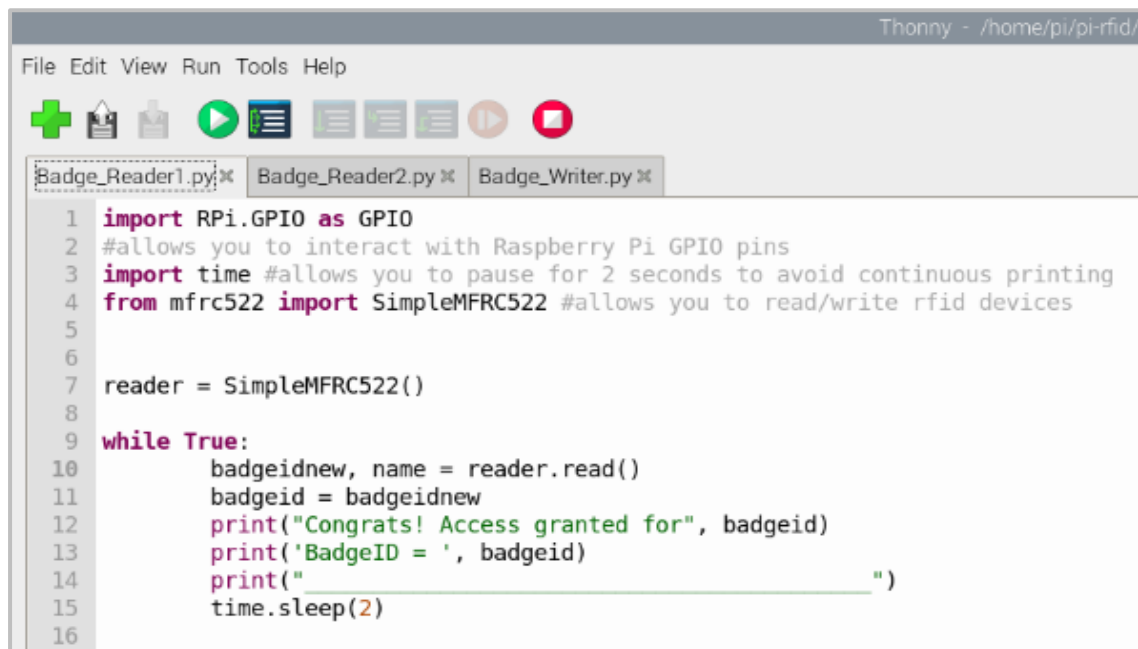
We ended up writing 3 python scripts: `Badge_Reader1.py`, `BadgeReader2.py`, and `Badge_Writer.py`. The inputs/outputs and a description of each script is shown in Table 1.

Essentially, `BadgeReader1.py` executes our original project proposal. `BadgeReader2.py` and `BadgeWriter.py` are both added functions to our proposed system.

All of our Python codes are shown in Figs. 3 - 5, to give the readers the ability to build on it if interested. Fig. 6 shows the final circuit assembly along with the pinout diagram. Note that heat sinks were installed as a precautionary measure to ensure proper operating temperature under all loads.

Table 1. Functions for the proposed system.

	Badge_Reader1.py	Badge_Reader2.py	Badge_Writer.py
Inputs	Key ring or smart card badge	Key ring or smart card badge	Key ring or smart card badge; Name string
Outputs	'Congrats, Access Granted' message to screen; Badge ID to screen	'Congrats, Access Granted' message to screen; Badge ID & name to screen; Current room occupants to screen	'Successfully Written!' message to screen; 'Your Badge ID is named:' message to screen
Description (What does it do)	Reads a smart card badge ID; Reads a key ring badge ID; Outputs a congrats message to screen; Outputs badge ID to screen	Reads a smart card badge ID; Reads a key ring badge ID; Outputs a congrats message to screen; Outputs badge ID, name, and current room occupants to screen	Writes a smart card badge ID and/or ring badge ID to contain the name of the user; Outputs 'Successfully Written' message & the new name of the badge



```

Thonny - /home/pi/pi-rfid/
File Edit View Run Tools Help
Badge_Reader1.py * Badge_Reader2.py * Badge_Writer.py *
1 import RPi.GPIO as GPIO
2 #allows you to interact with Raspberry Pi GPIO pins
3 import time #allows you to pause for 2 seconds to avoid continuous printing
4 from mfrc522 import SimpleMFRC522 #allows you to read/write rfid devices
5
6
7 reader = SimpleMFRC522()
8
9 while True:
10     badgeidnew, name = reader.read()
11     badgeid = badgeidnew
12     print("Congrats! Access granted for", badgeid)
13     print('BadgeID = ', badgeid)
14     print("_____")
15     time.sleep(2)
16

```

Fig. 3. BadgeReader1.py code.



```

Badge_Reader1.py * Badge_Reader2.py * Badge_Writer.py * SimpleMFRC522.py *
6
7 reader = SimpleMFRC522()
8
9 while True:
10     badgeidnew, name = reader.read()
11     start = time.clock()
12     badgeid = badgeidnew
13     print("Congrats! Access granted for", badgeid)
14     print('BadgeID = ', badgeid)
15     end = time.clock()
16     #print ("\nResponse Time:", end-start)
17     print("_____")
18     time.sleep(2)
19

```

Fig. 4. BadgeWriter.py code.

```

Thonny - /home/pi/pi-rfid
File Edit View Run Tools Help
Badge_Reader1.py x Badge_Reader2.py x Badge_Writer.py x
1 import RPi.GPIO as GPIO
2 #allows you to interact with Raspberry Pi GPIO pins
3 import time #allows you to pause for 2 seconds to avoid continuous printing
4 import os #allows you to print on a new line
5 from mfrc522 import SimpleMFRC522 #allows you to read/write rfid devices
6
7 reader = SimpleMFRC522()
8 name = '' #set name to blank
9 name_tot = '' #set name_tot, total people in the room, to blank
10
11 while True:
12     badgeidnew, namenew = reader.read()
13     badgeid = badgeidnew
14
15     #Only print name if it's a new person swiping (avoids
16     #the condition where badge is constantly displaying
17     #the tag ID)
18     a=1;
19     if namenew in name_tot:
20         a = 0
21         print('You already have access! Please enter')
22         print("_____")
23         time.sleep(2)
24
25     if (namenew != name and a!=0):
26         print("Congrats! Access granted for", namenew)
27         print('BadgeID = ', badgeid)
28         name_tot = name_tot + os.linesep + namenew
29         print("Already in the room: ", name_tot)
30         print("_____")
31         name = namenew
32         time.sleep(2)
33
34

```

Fig. 5. BadgeReader2.py code.

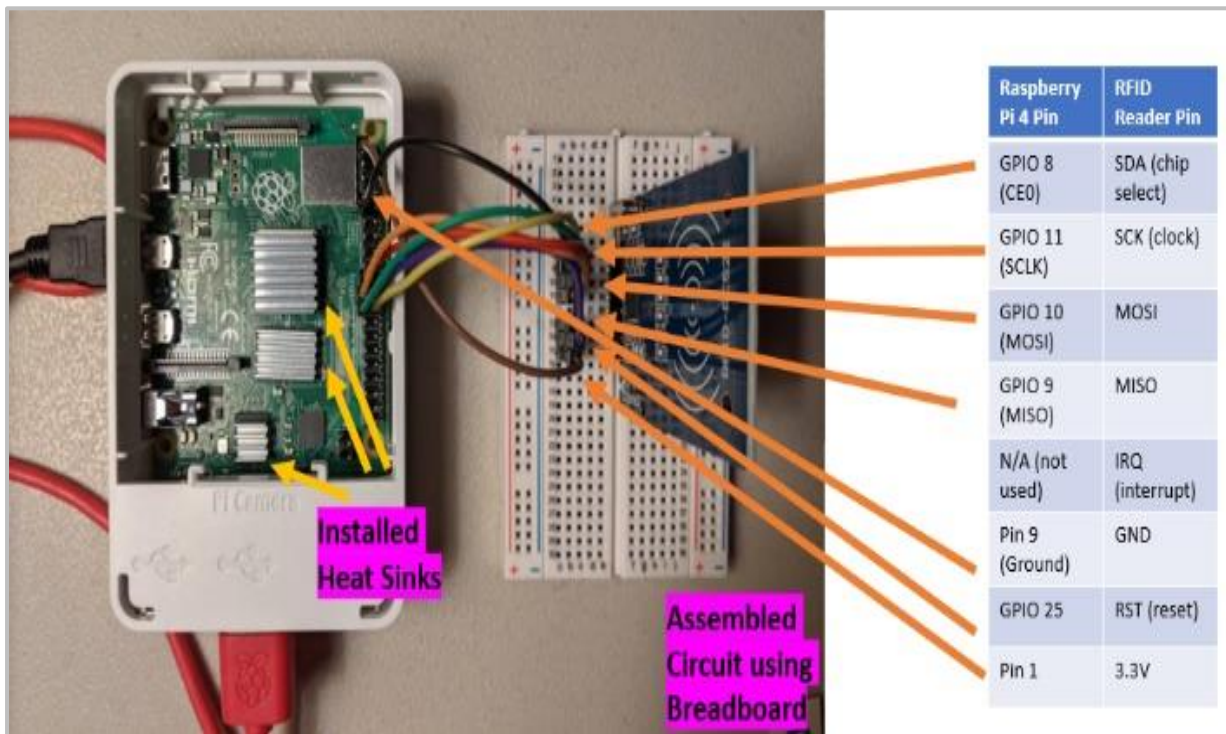


Fig. 6. Final circuit assembly and pinout diagram.



### 3. TESTING AND THE RESULTS OF THE PROPOSED SYSTEM

We ran through the test cases shown in section 2, and the results are indicated below by sub-bullet points:

- Amend badge reader script to utilize Python's 'time.clock()' function to accurately measure the time between the card being swiped and the display to the monitor. Fig. 7 shows how we amended BadgeReader1.py to accurately time the response to a card being read (see lines 12 and 15 that outputs the response time).
  - Using this script, the longest recorded response time was 0.006 seconds for all badges we swiped (Pass)
- Swipe 5 badges (from both architectures) and read the screen to ensure that at least four badges were read correctly. In order to ensure cards are read correctly, one must first ID the card, then make sure to look for that specific ID on the screen.
  - 5/5 Correct (Pass)
- Swipe a smart card chip badge and key ring badge, then check for the badge ID on the screen.
  - 2/2 Correct (Pass)
- Swipe a card within 0.25" from the reader and verify that the message ID appears on the screen
  - It Does (Pass)
- Swipe a card greater than 5" from the reader and verify that the message ID does NOT appear on the screen
  - It Does Not (Pass)
- Cool the environment down to 5 ° C. Check the temperature of the microprocessor by entering the command 'vcgencmd measure\_temp' into Linux. Once the temp reaches 5° C, scan a card over the reader and look for the ID on the screen. Repeat by heating the system to 27 ° C using a space heater and scan a card over the reader and look for the ID on the screen.
- Run the system repeatedly with an ambient environmental temperature of between 21-27° C. Run the system for 20 min, scanning several cards throughout the 20 min. Scan 10 cards in the last minute. (The goal is to heat the system up). Check the temperature of the microprocessor by entering the command 'vcgencmd measure\_temp' into linux.
  - Vcgencmd measure\_temp = 66.2° C (Pass, within required operating range because it's less than 80 ° C)

The system passed each of the above test cases, meaning that it has met all of its proposed features. It was able to identify the current occupants in the room which the user was swiping into, as well as associate a name with the badge ID, creating a friendlier and warm experience for the user. In the presented new system if the user has already swiped into the room and are re-entering, the system will recognize that as well, and tell them that they already have access and to please enter. The proposed system fits in the market a bit differently than the system in [3]. Instead of preventing 'buddy punching,' or making automatic changes to the user's environment, our proposed system provided other benefits, including allowing suppliers and OEMs to seamlessly share collaborative spaces, and ensuring older versions of badges don't become obsolete.

```

Thonny - /home/pi/pi-rfid
File Edit View Run Tools Help
Badge_Reader1.py x Badge_Reader2.py x Badge_Writer.py x SimpleMFRC522.py x

1 # Code by Simon Monk https://github.com/simonmonk/
2
3 from . import MFRC522
4 import RPi.GPIO as GPIO
5
6 class SimpleMFRC522:
7
8     READER = None
9
10    KEY = [0xFF,0xFF,0xFF,0xFF,0xFF,0xFF]
11    BLOCK_ADDRS = [8, 9, 10]
12
13    def __init__(self):
14        self.READER = MFRC522()
15
16    def read(self):
17        id, text = self.read_no_block()
18        while not id:
19            id, text = self.read_no_block()
20        return id, text
21
22    def read_id(self):
23        id = self.read_id_no_block()
24        while not id:
25            id = self.read_id_no_block()
26        return id
27
28    def read_id_no_block(self):
29        (status, TagType) = self.READER.MFRC522_Request(self.READER.PICC_REQIDL)
30        if status != self.READER.MI_OK:
31            return None
32        (status, uid) = self.READER.MFRC522_Anticoll()
33        if status != self.READER.MI_OK:
34            return None
35        return self.uid_to_num(uid)
36
37    def read_no_block(self):
38        (status, TagType) = self.READER.MFRC522_Request(self.READER.PICC_REQIDL)
39        if status != self.READER.MI_OK:
40            return None, None
41        (status, uid) = self.READER.MFRC522_Anticoll()
42        if status != self.READER.MI_OK:
43            return None, None
44        id = self.uid_to_num(uid)
45        self.READER.MFRC522_SelectTag(uid)
46        status = self.READER.MFRC522_Auth(self.READER.PICC_AUTHENT1A, 11, self.KEY, uid)
47        data = []
48        text_read = ''
49        if status == self.READER.MI_OK:
50            for block_num in self.BLOCK_ADDRS:
51                block = self.READER.MFRC522_Read(block_num)
52                if block:
53                    data += block
54            if data:
55                text_read = ''.join(chr(i) for i in data)
56
57        self.READER.MFRC522_StopCrypto1()
58        return id, text_read
59
60    def write(self, text):
61        id, text_in = self.write_no_block(text)
62        while not id:
63            id, text_in = self.write_no_block(text)
64        return id, text_in
65
66    def write_no_block(self, text):
67        (status, TagType) = self.READER.MFRC522_Request(self.READER.PICC_REQIDL)
68        if status != self.READER.MI_OK:
69            return None, None
70        (status, uid) = self.READER.MFRC522_Anticoll()
71        if status != self.READER.MI_OK:
72            return None, None
73        id = self.uid_to_num(uid)
74        self.READER.MFRC522_SelectTag(uid)
75        status = self.READER.MFRC522_Auth(self.READER.PICC_AUTHENT1A, 11, self.KEY, uid)
76        self.READER.MFRC522_Read(11)
77        if status == self.READER.MI_OK:
78            data = bytearray()
79            data.extend(bytearray(text.ljust(len(self.BLOCK_ADDRS) * 16).encode('ascii')))
80            i = 0
81            for block_num in self.BLOCK_ADDRS:
82                self.READER.MFRC522_Write(block_num, data[(i*16):(i+1)*16])
83                i += 1
84            self.READER.MFRC522_StopCrypto1()
85            return id, text[0:(len(self.BLOCK_ADDRS) * 16)]
86
87    def uid_to_num(self, uid):
88        n = 0
89        for i in range(0, 5):
90            n = n * 256 + uid[i]
91        return n

```

Fig. 7. The amended BadgeReader1.py that includes time.clock().

#### 4. CONCLUSIONS

An embedded system which reads an RFID chip within a key ring or smart card badge was presented in this paper. The proposed system worked as follows: when the user swiped an RFID device, independent of the RFID architecture (key ring or smart badge), the badge ID was displayed on the monitor, along with a welcome message. The proposed design was also able to identify the current occupants in the room which the user is swiping into, as well as associate a name with the badge ID, creating a friendlier and warm experience for the user. The proposed system was also able to identify if the users have already swiped into the room and re-entered. The system was able to recognize that as well, and tell them that they already have access and ask them to enter.

Finally, what distinguishes our system from others is that it is more open source and allows for several badge architectures to be read. Also, the features that we have added to this proposed system to identify the occupants in the room by name sets our project apart from any of the badge readers that we could find in the current state-of-the-art.

#### REFERENCES

- [1] C. Corches, O. Stan, L. Miclea, M. Daraban, "Embedded RTOS for a smart RFID reader," *IEEE 25th International Symposium for Design and Technology in Electronic Packaging*, pp. 220-223, 2019.
- [2] M. Ougida, H. Yamaguchi, T. Higashino, "Trajectory-assisted robust RFID-tagged object tracking and recognition in room environment," *The 23rd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, New York, NY, USA, pp. 11-15, 2020.
- [3] Crown Security Products, "CR-C2 fingerprint time clock w/contactless RFID technology," *Crown Security Products*, 2020. <<https://crownsecurityproducts.com/time-clocks/fingerprint-time-clocks/cr-c2-contactless-biometric-time-clock-rfid.html>>
- [4] AlphaCard LLC, "Employee access control systems," *AlphaCard LLC*, 2022. <<https://www.alphacard.com/id-card-solutions/employee-access-control>>
- [5] C. Corches, O. Stan, L. Miclea, "Embedded RTOS for a smart RFID reader," *2019 IEEE 25th International Symposium for Design and Technology in Electronic Packaging*, pp. 220-223, 2019.
- [6] The Raspberry Pi Foundation, "SPI," *Raspberry Pi*. <<https://www.raspberrypi.org/documentation/hardware/raspberrypi/spi/README.md>>
- [7] SunFounder, "RFID kit mifare RC522 RFID reader module with S50 white card and key ring for Arduino Raspberry Pi," *Amazon*. <[https://www.amazon.com/dp/B07KGBJ9VG?psc=1&ref=ppx\\_yo2\\_dt\\_b\\_product\\_details](https://www.amazon.com/dp/B07KGBJ9VG?psc=1&ref=ppx_yo2_dt_b_product_details)>
- [8] E. Young, M. Macdonald-Wallace, "MFRC522-python," *GitHub*, 2019. <<https://github.com/pimylifeup/MFRC522-python>>
- [9] CanaKit Corporation, "Sandisk pre-installed NOOBS MircroSD cards," *CanaKit Corporation*. <<https://www.canakit.com/raspberry-pi-sd-card-noobs.html>>