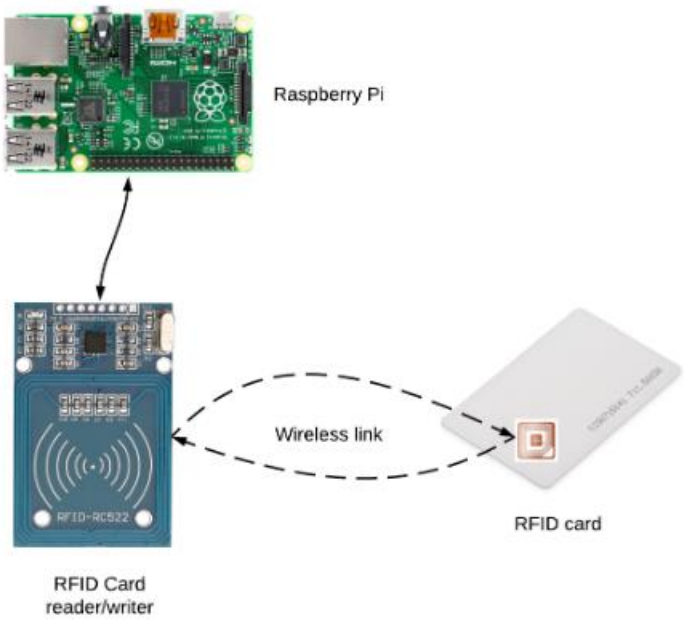
RFID CARDS

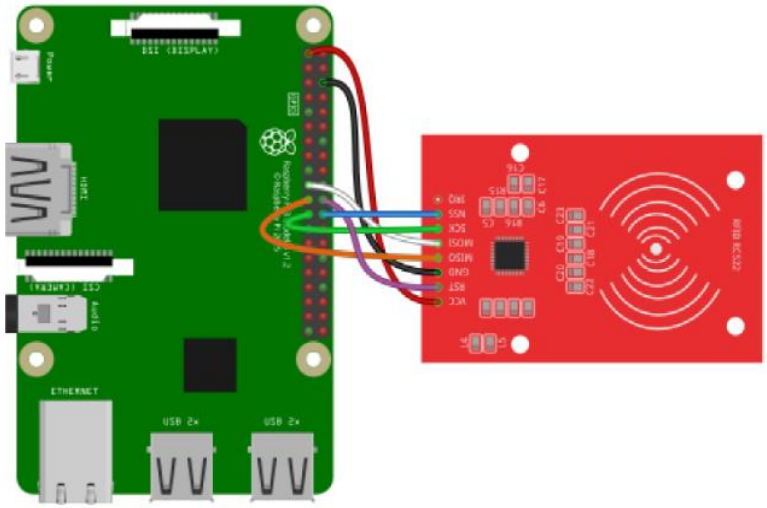
Objective: TO USE RFID CARDS WITH A RASPBERRY PI

Abstract:

Radio frequency identification (RFID) devices are an essential part of almost all physical security systems. RFID cards and card readers are used to restrict access to buildings or rooms. In this context, the RFID card stores a unique identification number that is wirelessly detected by a RFID card reader attached to a wall. If the RFID card's identification number matches an identification number from a stored list of authorized cards, the door is unlocked and the cardholder can enter the room. RFID devices operate at high-frequency from 3MHz to 30MHz. But most RFID devices usually operate at 13.56MHz.



Basic components of a Raspberry Pi RFID system



ALGORITHM:

1)Connect RFID reader to the Raspberry Pi.

- Ensure stable power supply for both devices.

2)Software Setup:

- Install necessary RFID reader libraries (e.g., pi-rc522).
- Install any additional required software for data processing.

3)RFID Reading and Processing:

- Initialize the RFID reader and configure GPIO pins.
- Continuously monitor for card detection.

4)Output or Action:

- Display messages or trigger actions based on the validation result.

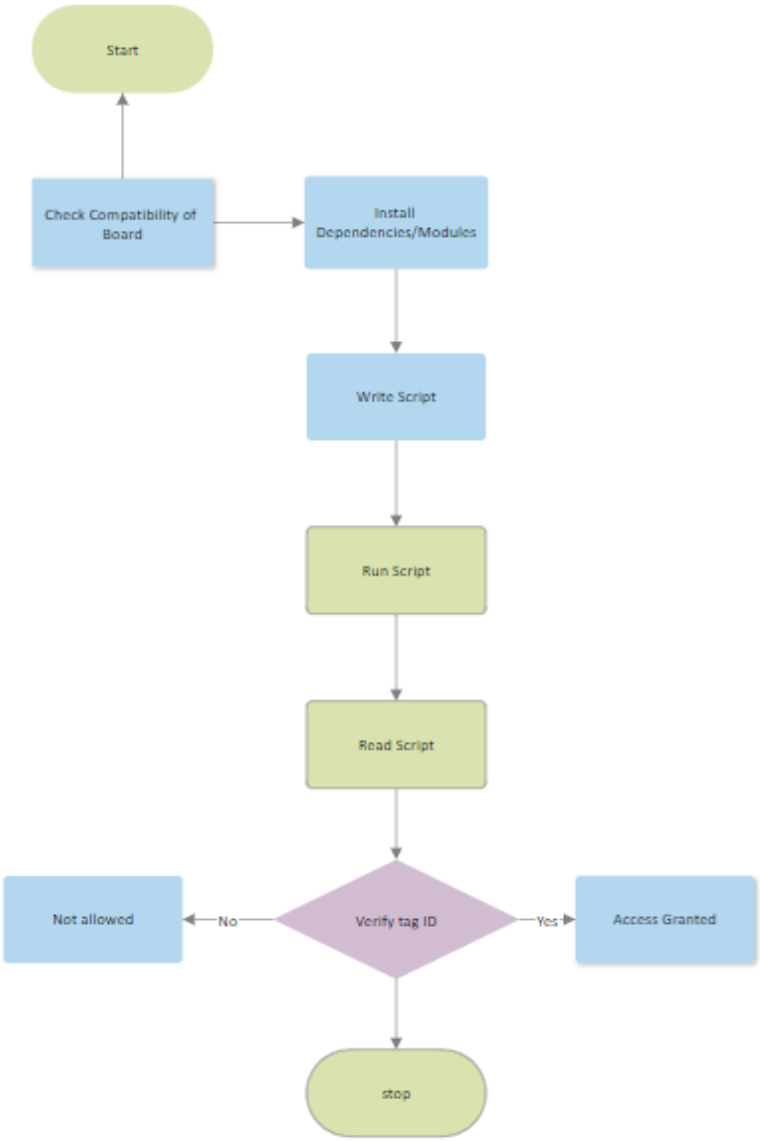
5)Testing and Deployment:

- Test the system with various RFID cards for accuracy.
- Deploy the system securely in the desired environment.

6)Maintenance and Updates:

- Regularly maintain the system and check for issues.
- Update the system for new features or security patches.

FLOWCHART:


Start
Check Compatibility of Board
Install Dependencies/Modules
Write Script
Run Script
Read Script
Not allowed
No
Verify tag ID
Yes
Access Granted
stop

CODE TO READ THE RFID CARD

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

rfid = SimpleMFRC522()

while True:
    id, text = rfid.read()
    print(id)
    print(text)
```

Our first step will be to read the information on the RFID card. To do this, connect the components as shown in the diagram above. Then, create a Python file `rfidReader.py` and paste the following code.

First, we import the necessary modules. Then we use `rfid = SimpleMFRC522()` to create a new object which we will call RFID. To read the RFID card date, we call the read() function and store its output in the variables `id` and `text`.
Finally, we print the ID and text of the card to the terminal with `print(id)` and `print(text)`, respectively.

CODE TO WRITE TO THE RFID CARD

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

rfid = SimpleMFRC522()

try:

        print("Hold tag near the module...")

        rfid.write("Circuit basics")

        print("Written")

finally:

        GPIO.cleanup()
```

To write information on the RFID card, we use another method of the RFID object: `rfid.write()`. We can see how this works in the code below. We create a file called `rfidWriter.py` and paste the code below.
A notable difference between `rfidReader.py` and `rfidWriter.py` is that we do not have a while loop in the latter. This is mainly because we do not want to write the same information on different cards. So we execute the code above as many times as we want to write to different cards. Everything else stays the same, except that we invoke the `rfid.write("Circuit basics")` when we bring the card closer to the RFID writer. In this case, we write the string `Circuit basics` to the card.

CODE DESCRIPTION

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

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(17, GPIO.OUT)
rfid= SimpleMFRC522()
channel = 17

def relay_on(pin):
   GPIO.output(pin,GPIO.HIGH)

def relay_off(pin):
   GPIO.output(pin,GPIO.LOW)

while True:
     id, text = rfid.read()
     print(id)

      if id == 1002059512185:
       relay_on(channel)
       print(text+":Access granted")
       time.sleep(5)
       relay_off(channel)

     else:
       relay_off(channel)
       print("Not allowed...")
```

Instead of using the LED library to control the GPIO, we declare GPIO pin 17 as an output (`GPIO.setup(17, GPIO.OUT)`) and we write two functions that turn ON and OFF our output pin: `relay_on(pin)` and `relay_off(pin)`, respectively. We modify the original `rfidReader.py` to constantly check the ID of each detected card using the line `if id == 1002059512185`. If the ID of the detected card matches with the ID in our program, we switch ON the relay; otherwise, we keep it OFF.

REAL TIME CONSTRAINT:

A significant real-time constraint in RFID using Raspberry Pi is the processing delay caused by complex data analysis algorithms. When dealing with a large number of RFID tags and complex data processing requirements, the Raspberry Pi's processing power might become a limiting factor. For instance, in a high-speed inventory management system where multiple RFID tags are read simultaneously, the Raspberry Pi's processing capabilities might not be sufficient to analyze and respond to the data in real-time. This processing delay can lead to inaccuracies in inventory tracking and control, potentially impacting the overall operational efficiency and productivity of the system. Therefore, optimizing the data processing algorithms and considering the limitations of the Raspberry Pi's processing power are crucial to meet the real-time constraints of the RFID system

CONCLUSION:

Implementing RFID (Radio Frequency Identification) technology using Raspberry Pi offers an array of benefits for various applications. By combining the capabilities of RFID with the flexibility and accessibility of the Raspberry Pi, a cost-effective and efficient system can be created for tracking, monitoring, and managing assets, inventory, or access control.