

Vaccine verification using RFID based secure authentication

30th September 2021

teambiOs

1. Nidhin S Naushad
2. Athul Menon
3. Ranit Pradhan
4. Aswin C

OVERVIEW

As the world is bouncing back up from the COVID-19 pandemic, universities, companies, and businesses are in urgent need of a simple, reliable, and trustworthy way to check whether the people are fully vaccinated and in good health in order to ensure the safety of themselves and others. Currently, we depend upon cheap digital thermometers and other tools which are often unreliable and hard to manage when dealing with universities and schools.

Body temperature verification and checking if he/she has taken the vaccine have become a necessity in all domains. Providing a safe and instantaneous way to check if someone is vaccinated using RFID and its analogous technologies is the goal of this project. The project uses RFID technology for authentication and also implements technology for the verification of body temperature without contact.

A detailed technical description of the project with pictures of the hardware from each stage as proof of work is the overall substance of this report.

OBJECTIVE

1. RFID-based secure authentication.
2. Contactless temperature verification.
3. Storing data associated with the vaccination...

REQUIRED COMPONENTS

- RFID MFRC-5222

-
- Temperature sensor MLX90614
 - Arduino UNO
 - ESP 32 Development Board
 - Bread Board
 - Resistors
 - Connection Wires
 - LED / Buzzer

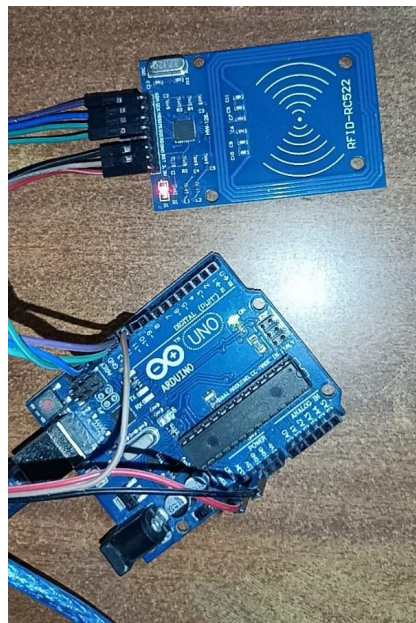
MILESTONES

RFID authentication

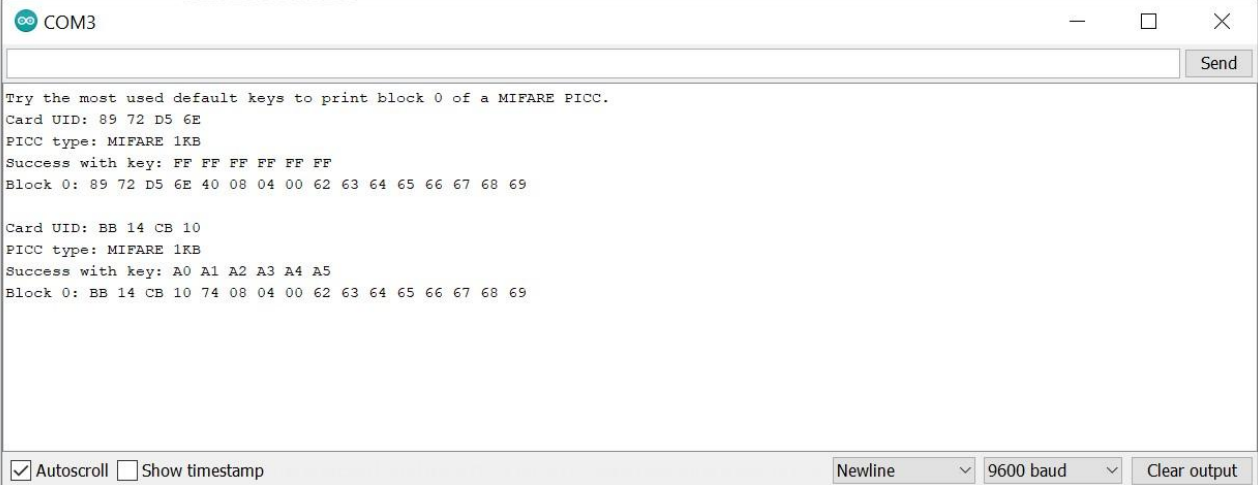
Basic RFID authentication is done in two steps. Firstly, the connection with the hardware is made by the Arduino Uno with MFRC 522 and it is checked whether it is working properly by testing with the default keys provided by the manufacturers themselves. For verification, we use `rfid_default_keys` code from the custom library of RC522.

Further with the help of the sample code-named `rfid_write_personal_data` from MFRC 522's custom library, we will encode the RFID tags with our required data for authentication. In the first phase, the input encoded into the tags will be student name and vaccine status, in the third phase, we'll automate this process. Further using `rfid_read_personal_data` from MFRC 522's custom library, we'll verify these details.

Hardware:



Result:



```
COM3

Try the most used default keys to print block 0 of a MIFARE PICC.
Card UID: 89 72 D5 6E
PICC type: MIFARE 1KB
Success with key: FF FF FF FF FF FF
Block 0: 89 72 D5 6E 40 08 04 00 62 63 64 65 66 67 68 69

Card UID: BB 14 CB 10
PICC type: MIFARE 1KB
Success with key: A0 A1 A2 A3 A4 A5
Block 0: BB 14 CB 10 74 08 04 00 62 63 64 65 66 67 68 69

☒ Autoscroll ☐ Show timestamp
Newline 9600 baud Clear output
```

Contactless temperature measurement

Contactless temperature measurement is done by integrating Arduino UNO with MLX 90614 non-contact human body infrared temperature sensor with help from the Arduino standard library to support Melexis MLX90614 infrared thermometer using the I2C interface. For I2C support, we plan to use the SoftWire software bit-banging Arduino library. Being GNU LGPL v2.1 and open source, this library is compatible with all architectures, hence after checking it with Arduino UNO it can also be used with a Wi-Fi module. For implementation, we are using mlxtest code which is a library example for the MLX90614 Temp Sensor from the Adafruit_MLX90614 file.

From the first two sections, we get the data of the RFID verification and the temperature measurement. Using conditional statements in the code, we use the LED or a buzzer to signal whether the student can enter or not.

Code:

```
#include <Adafruit_MLX90614.h>
Adafruit_MLX90614 mlx = Adafruit_MLX90614();
void setup() {
  Serial.begin(9600); while (!Serial);

  Serial.println("Adafruit MLX90614 test");

  if (!mlx.begin()) {
    Serial.println("Error connecting to MLX sensor. Check wiring.");
    while (1);
  };
};
```

```

    Serial.print("Emissivity = "); Serial.println(mlx.readEmissivity());
    Serial.println("=====");
}

void loop() {
    Serial.print("Ambient = "); Serial.print(mlx.readAmbientTempC());
    Serial.print("*C\tObject = "); Serial.print(mlx.readObjectTempC());
    Serial.println("*C");
    Serial.print("Ambient = "); Serial.print(mlx.readAmbientTempF());
    Serial.print("*F\tObject = "); Serial.print(mlx.readObjectTempF());
    Serial.println("*F");

    Serial.println();
    delay(500);
}

```

Hardware:



Result:

Ambient = 31.03°C	Object = 37.59°C
Ambient = 87.85°F	Object = 99.66°F

Establishing server-side architecture

We prototyped the server-side architecture by using a Nodejs backend connecting to a MongoDB database, which is suitable for a large number of simultaneous reads and writes, that will store the relevant information required for the authentication in a light and fast database. MongoDB also offers support for JSON and other data formats which might help in future expansion and extension of the project into other areas. We are currently using the database to store and retrieve the information received from the RFID and temperature sensor along with the time it was received. This data also has the potential to be used as a time series dataset in a variety of applications such as predicting the peak times, potential crowding, etc.

Integration with Co-WIN API

Co-WIN Public APIs allow any third-party application to access certain un-restricted information that can be shared with its users. We used the newly released 'Know Your Customer's/Client's Vaccination Status' or KYC-VS to access the vaccination status of a person. To do so, the individual's registered mobile number and name are required, which we obtain from the information on the RFID to generate a one-time password or OTP. The two-layer authentication will prompt CoWin to respond on the vaccination status—0 if the person is not vaccinated, 1 if the person is partially vaccinated and 2 if the person is fully vaccinated. This report will be digitally signed and can be shared instantly by the verifying entity. Then, this is stored on the server till it is updated again.

Testing

We tested and debugged the entire application. First, we checked if the RFID tags are properly overwritten with our authentication information by physically reading it with MFRC 522 and verifying it using Arduino CLI. Further, we then checked the working of contactless temperature sensors. Finally, we tested the security and performance of the deployment of the database and backend.