week-4

Aim: write a Program to implement RFID using Arduino.

SPI Protocol:

→ SPI is a serial Peripheral interfate.

→ It is a serial communication Protocol often used in embedded systems for high-speed data exchanges between devices on the bus.

→ It coperates using master-slave.

4 signals.

1. a clock (SSLK)
2. a master output/slave input (MOSI)
3. a master input/slave output (MISO)
4. a slave select (SS)

→ The master Pulls low on a slaves's line to

→ It supports full duplex communication.

→ when master and slave can transmit the data simultaneously.

→ The data. speed reaches to 100 MHZ.

→ The first line shows the firm wave version of the MFRC522 IC.

→ In this case, the result is 0x92

→ A typical 1K RFID tag has 1K byte of memory organised into 16 sectors each sector consists of 4 blocks.

Sector	Block	Byte Number within a Block																Descrip
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
15	3	Key A						Access Bits				Key B.						Sector Tra
	2																	Data
	1																	Data
	0																	Data.
14	3	Key A						Access Bits				Key B						Sector Trailer
																		Data
																		data
																		Data.

→ Block 0 of sector 0 is reserved for storing manufacturer Data.

→ It contains 4 bytes unique ID.

→ each sector consists of three Data blocks.

→ which can be used for storing user data and this is known as sector Trailer.

→ 16 sector contains 16 sector Trailers.

→ And each sector Trailer consists of. Key A 6 bytes mandatory

Key B optional - 6 bytes.

4 bytes for access Bits.

Hardware Requirements:

- → Arduino
- → Jumper wires.
- → USB cable
- → RC522 RFID reader
- → card
- → Key chain.

RST	9
SDA(SS)	10
MOSI	11
MISO	12
SCK	13

Program (demorfid.ino)

```
# include <SPI.h>
# include <MFRC522.h>
# define RST_PIN 9
# define SS_PIN 10
MFRC522 mfrc522 (SS_PIN, RST_PIN);
void setup(){
    Serial.begin (9600);
    while (! serial);
    SPI.begin ();
    mfrc522.PCD_Init();
    delay (4);
    mfrc522.PCD_DumpVersionToSerial();
  Serial.Println(F("Scan PICC to see UID, SAK, type and
                                     data blocks ..."));
}
```

```
void loop(){
    if (!mfrc522.PICC_IsNewCardPresent()){
        return;
    }
    if (!mfrc522.PICC_ReadCardSerial()){
        return;
    }
    mfrc522.PICC_DumpToSerial(&(mfrc522.uid));
}
```

rfid_read_Personal_data.ino.

```
#include <SPI.h>
#include <MFRC522.h>
#define RST_PIN    9
#define SS_PIN     10
MFRC522 mfrc522(SS_PIN, RST_PIN);
void setup(){
   Serial.begin(9600);
   SPI.begin();
  mfrc522.PCD_Init();
  Serial.Println(F("Read Personal data on a MIFARE PICC:"));
}
```

```
Void loop() {

  MFRC522::MIFARE_Key Key;

  for (byte i=0; i<6; i++) Key. KeyByte[i] = 0xFF;

  byte block;
  byte len;

  MFRC522::StatusCode status;

 if (! mfrc522. PICC_IsNewCardPresent()){

     return;
  }
 if (!mfrc522. PICC_ReadCardSerial()) {

     return;
  }

Serial.Println(F("** Card Detected:**"));

mfrc522.PICC_DumpDetailsToSerial(&(mfrc522.uid));

Serial.Print(F("Name: "));

byte buffer1[18];

block =4;

len =18;
status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_
                AUTH_Key_A, 4, &Key, &(mfrc522.uid));

if (status != MFRC522::STATUS_OK) {
      Serial.Print(F("Authentication failed: "));
     Serial.Println(mfrc522.GetStatusCodeName(status));
     return;

 }

status = mfrc522.MIFARE_Read(block, buffer1, &len);
if (status != MFRC522::STATUS_OK) {
     Serial.Print(F("Reading failed: "));
    Serial.Println(mfrc522.GetStatusCodeName(status));
    return;
}.
  for (uint8_t i = 0 ; i<16; i++) {
        Serial.write(buffer2[i]);
   }
 Serial.Println(F("/n** End Reading**/n"));

delay(1000);
mfrc522.PICC_HaltA();
mfrc522.PCD_stopCrypto();
}.
```

Write:

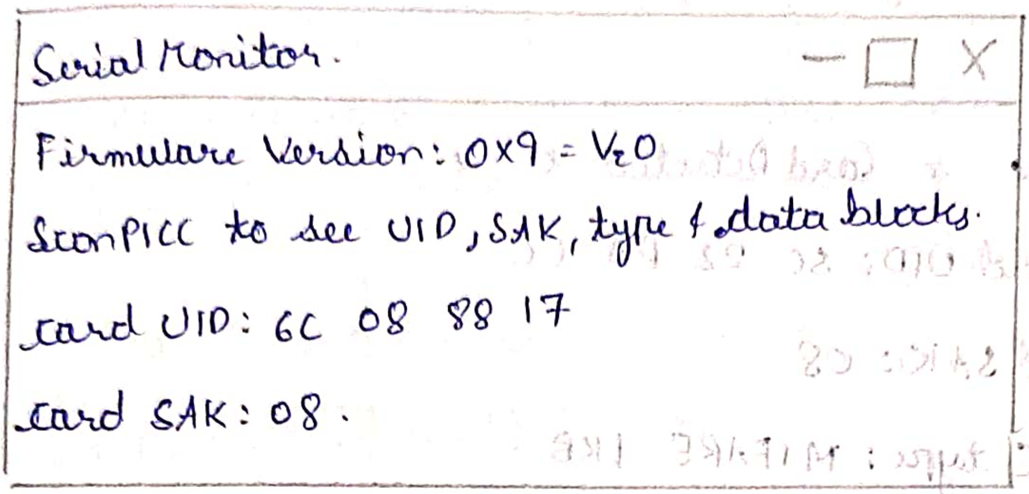
```
# include < SPI. h>
# include< MFRC522. h>
# define RST_PIN 9
# define SS_PIN 10.
MFRC522 mfrc522 ( SS_PIN, RST_PIN);
void setup() {
    Serial.begin(9600);
    SPI. begin();
    mfrc522:PCD_Init();
    Serial.Println (F ("write Personal data on a MIFARE
                                                    PICC"));
 }
 void loop() {
    MFRC522:: MIFARE_Key Key;
    for (byte i=0; i<6; i++)
      Key-KeyByte[i] = 0XFF;
    if (! mfrc522.PICC_IsNewCardPresent ()) {
        return;
    }
  if (! mfrc522.PICC_ReadCardSerial ()) {
        return;
  }
  Serial.Print (F("Card UID:"));
 for (byte i=0; i< mfrc522.uid. size; i++) {
   Serial.Print(mfrc522.uid. uid ByteCi]<0x10? "0":" ");
   Serial. Print (mfrc522. uid. uid Byte [i], HEX);
}
```

```
Serial.Print (F ("PICC type:"));
MFRC522:: PICC_Type PICC Type = mfrc522.PICC_Get
            Type (mfrc522.uid.sak);
Serial.Println(mfrc522.PICC_Get Type Name (Picc.Type));
byte buffer[34];
byte block;
MFRC522:: statuscode status;
byte len;
Serial.set Timeout (20000 L);
Serial.Println (F ("Type Firstname, ending with #"));
len = Serial.read Bytes until ('#', (char*) buffer, 20));
for (byte i = len; i<20; i++) buffer [i] = ' ';
block = 1;
status = mfrc522.PCD_Authenticate (MFRC522::
            PICC_CMD_MF_AUTH_KEY_A, block, 4 Key,
            &(mfrc522.uid));
if (status != MFRC522:: STATUS_OK) {
Serial.Print (F ("PCD_Authenticate () failed: "));
Serial.Println (mfrc522. GetStatusCodeName (status));
 return;
}
status = mfrc522.MIFARE_Write (block, buffer, 16);
if (status != MFRC522:: STATUS_OK) {
 Serial.Print (F ("MIFARE_Write () failed: "));
 Serial.Println (mfrc522. Get Status Code Name (status));
          return;          }
```

```
else Serial.Println (F("MIFARE_Write() success: "));
block = 2;
Status = mfrc522.PCD_Authenticate (MFRC522::PICC_
CMD_MF_AUTH_KEY_A, block, &key, & (mfrc522.uid));
if (status != MFRC522:: STATUS_OK ){
    Serial.Print (F ("PCD_Authenticate () failed: "));
    Serial.Println (mfrc522. GetStatus Code Name (status));
    return;
 }
 Status = mfrc522. MIFARE_Write (block, &buffer [16], 16);
if (Status != MFRC522:: STATUS_OK){
  Serial.Print (F ("MIFARE_write () failed: "));
Serial.Println (mfrc522. Get Status Code Name (status));
return;
}
else Serial.Println (F( "MIFARE_Write() Success: "));
 Serial.Println (" ");
mfrc522.PICC_HaltA();
mfrc522.PCD_StopCrypto1();
}.
```

Output:

Dump info:



```
Serial Monitor.

Firmwlare Version: 0x9 = V2.0

ScanPICC to see UID, SAK, type & data blocks.

card UID: 6C 08 88 17

card SAK: 08.
```

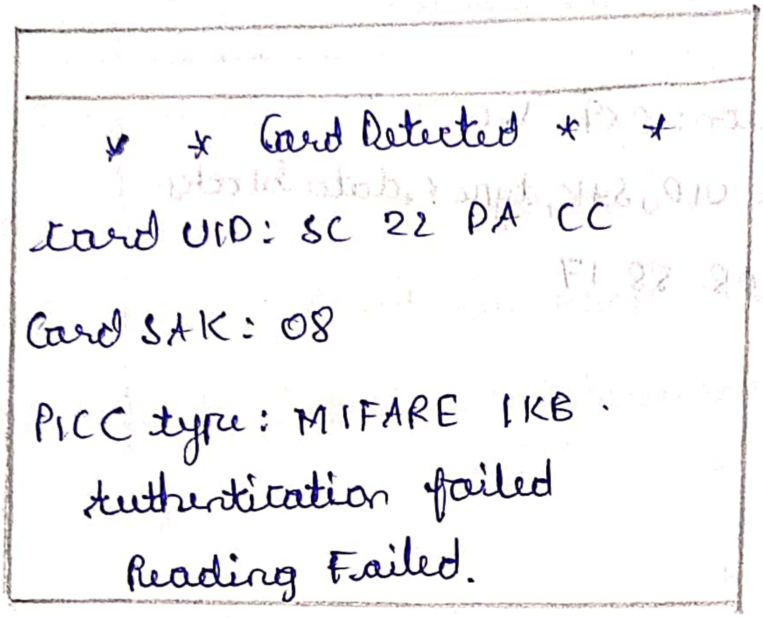
PICC_type: MIFARE1KB.

Sector	Block	0	1	2	3	4	5	6	7	8	9	10	11	12
15	63	00	00	00	00	00	00	00	00	00	00	00	00	00
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
0	0	61	08	88	17	68	89	62	64	65	66	67	68	69

Access Bits:
[0 0 1]
⋮
[0 0 0]

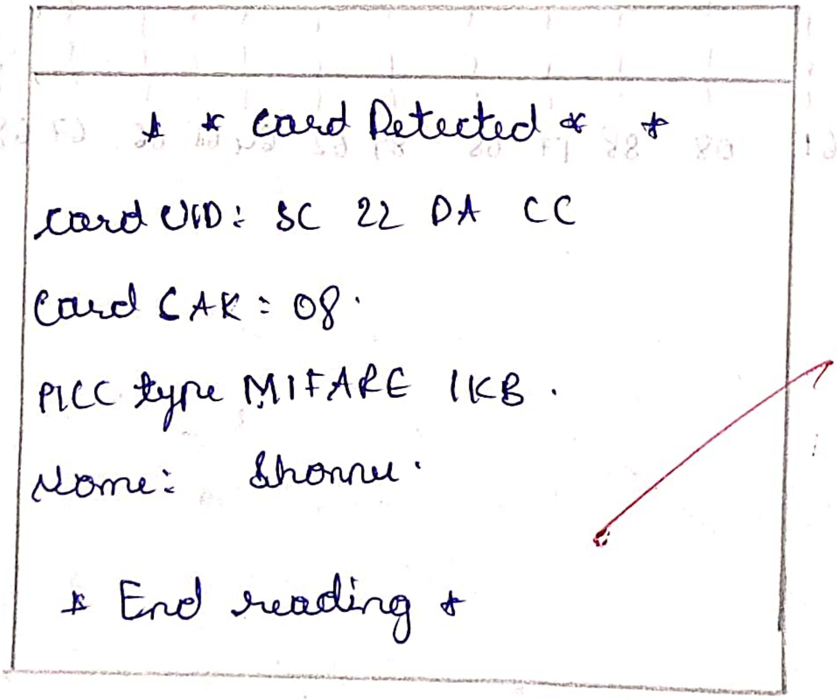
Read -> output.

After reading the card.



```
   * * Card Detected * *
card UID: SC 22 DA CC
Card SAK: 08
PICC type: MIFARE 1KB.
  Authentication failed
    Reading Failed.
```

Writing data on a MIFARE PICC then reading the data.



```
   * * card Detected * *
card UID: SC 22 DA CC
Card CAK: 08.
PICC type MIFARE 1KB.
Name:   Shonne.

  * End reading *
```

write:

write personal data on a MIFARE PICC

Card UID: 5C 22 DA CC PICC type:
MIFARE 1KB.

Type First name, ending with #

PCD_Authenticate() success:

MIFARE_Write() success:

MIFARE_write() success (Arduino #)

Type first name ending with #

MIFARE_write() success:

MIFARE_write() success: (~~Sruth~~ Shonnu #)

PICC: Proximity integrated circuit cards that serves as different electromagnetic field coupling between reader and the card.

PCD: Proximity Coupling device Also Known as RFID. They decode the RFID tags and communicate with them based on ISO 14444 3 standard.

PCD can Perform read and write operation of data.

PCD ensures the generation of a magnetic field, whereas the antena of PICC allows receiving the magnetic field.

=> The frequency generated by MFRC522 is 13.56 MHZ

=> The PICC_HaltA() function sends a halt command to the RFID card which stops further communication with Card.

=> The PCD_stop Crypto() function stops the encryption of the data between the RFID card and the reader.