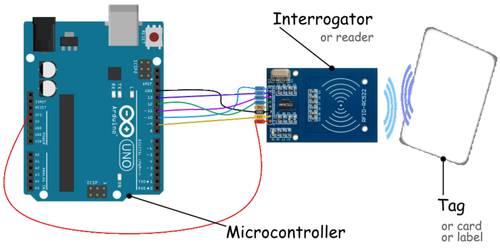
Transacting using RFID System

Prerequisite /priːrɛkwɪzɪt/
Before transaction, we should have written to the PICC a number like 5000 which represents money.

Connection Diagram



Code:
The code below does few things:
- Reading the initial Balance from the RFID card
- if the initial balance is more than the set transport fare then deduct set transport fare from the initial balance
- Save the new balance to the RFID card

```
#include <SPI.h>
#include <MFRC522.h>
#define RST_PIN 9
#define SS_PIN 10

MFRC522 mfrc522(SS_PIN, RST_PIN);
MFRC522::MIFARE_Key key;
MFRC522::StatusCode card_status;

void setup(){
  Serial.begin(9600);
  SPI.begin();
  mfrc522.PCD_Init();
  Serial.println(F("TRANSACTION:"));
}

void loop(){
  byte block_number = 4;
  byte buffer_for_reading[18];
  for (byte i = 0; i < 6; i++){
    key.keyByte[i] = 0xFF;
  }

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


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

  String initial_balance = readBalanceFromCard(block_number, buffer_for_reading);
  operateData(block_number, initial_balance);

  mfrc522.PICC_HaltA();
  mfrc522.PCD_StopCrypto1();
}

String readBalanceFromCard(byte blockNumber, byte readingBuffer[]){
  card_status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, 4, &key, &(mfrc522.uid));

  if(card_status != MFRC522::STATUS_OK){
    Serial.print(F("Authentication failed: "));
    Serial.println(mfrc522.GetStatusCodeName(card_status));
    return;
  }

  byte readDataLength = 18;
  card_status = mfrc522.MIFARE_Read(blockNumber, readingBuffer, &readDataLength);

  if(card_status != MFRC522::STATUS_OK){
    Serial.print(F("Reading failed: "));
    Serial.println(mfrc522.GetStatusCodeName(card_status));
    return;
  }

  String value = "";
  for (uint8_t i = 0; i < 16; i++){
    value += (char)readingBuffer[i];
  }
  value.trim();

  return value;
}

bool saveBalanceToCard(byte blockNumber, byte writingBuffer[]){
  card_status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, blockNumber, &key, &(mfrc522.uid));

  if(card_status != MFRC522::STATUS_OK){
    Serial.print(F("PCD_Authenticate() failed: "));
    Serial.println(mfrc522.GetStatusCodeName(card_status));
    return;
  }
  else{
    //Serial.println(F("PCD_Authenticate() success: "));
  }

  // Write block

  card_status = mfrc522.MIFARE_Write(blockNumber, writingBuffer, 16);
  if(card_status != MFRC522::STATUS_OK){
    Serial.print(F("MIFARE_Write() failed: "));
    Serial.println(mfrc522.GetStatusCodeName(card_status));
    return;
  }
  else{
    //Serial.println(F("Data saved."));
    delay(5000);
    return true;
  }
}

void operateData(byte blockNumber, String initialBalance){
  int fareToKimironko = 450;
  float newBalance = initialBalance.toInt()-fareToKimironko;

  if(initialBalance.toInt()<fareToKimironko){
    Serial.print("Insufficient Balance: ");
    Serial.println(initialBalance);
    return;
  }

  String initial_balance_str;
  char writingBuffer[16];
  initial_balance_str = (String)newBalance;
  initial_balance_str.toCharArray(writingBuffer, 16);
  int strLeng = initial_balance_str.length()-3;
  /*
   * This servers to add spaces to the typed text in order to fill up to 16 characters
  */

  for(byte i = strLeng; i < 30; i++){
    writingBuffer[i] = ' ';
  }

  Serial.println("\n********************");
  Serial.println("Processing...");
  if(saveBalanceToCard(blockNumber, writingBuffer)==true){
    Serial.print("Initial Balance: ");
    Serial.println(initialBalance);
    Serial.print("Fare to Kimironko: ");
    Serial.println(fareToKimironko);
    Serial.print("New Balance: ");
    Serial.println(newBalance);
    Serial.println("Transaction Succeeded.\n");
  }
  else{
    Serial.print("Transaction Failed.\nPlease try again.\n");
  }
  Serial.println("********************\n");
}
```