A PROJECT REPORT BASED ON IOT

“SMART BILLING SYSTEMS FOR WATER SUPPLIERS”

TEAM “TRIS”

Submitted by-

CH. MANASVINI ABHIGNA

KANTAREDDY VARSHITHA

B.RUTHVIKA RATHOD

Under the guidance of



# ABSTRACT:

Extensive use of various types of IoT devices has resulted in high demand for developers with IoT skills. This document proposes the Smart Billing System for water suppliers through Internet of Things (IoT) platform. This module is to indicate the water usage and automatic control of water flow and also prepaid water billing via IoT for water suppliers. It is to provide smart billing scheme for water suppliers and board for automatic the control and monitoring water usage system with no water wastage . Wastage of water in process of manually operated water pump, human error associated with manually operated system, improper bill creation, delay in bill availability and delay in payments process are problems that we face to overcome this process we mean this project helps a lot.

PROJECT STATEMENT:

Now a day's metropolitan cities operates water tanker service for delivery to residents needing drinking water from several fill stations across cities. Water tankers or private (lorry) operators gets registered with these filling stations for facilitating water distribution to citizens. Once they get registered, a card is issued to them which is used for payments. User has the facility to top-up their card through web application. Each fill station is equipped with one more hand-held device (based on number of pumps in the fill station). Handheld device has facility to read/write into a RFID based smart card as well as WiFi modem to communicate with a central server.

Water tanker supplier, who visits a fill station, has to tell the requirements before arriving the fill stations. This can be done through Web app (like capacity of water required, area to be delivered) which will be saved in the central server. Once the operator reaches fill station he fills the water and swipes the card to the handled device so that the amount gets deducted from the card. If the amount is insufficient in the card a message is displayed on a screen.

**WORKING PROCESS**:

1. **Arduino IDE** must be installed in the system along with all the required libraries.

The required **libraries** here are:

1. SPI.h
2. MFRC522.h
3. Servo.h
4. ESP8266WiFi.h
5. PubSubClient.h
6. These are the following connections that must be connected to the **Node MCU ESP8266 Module :**

**RFID Reader Node MCU**

Gnd Gnd

Vcc 3V3

SDA(SS) D8

RST D3

SCK D5

MOSI D7

MISO D6

**FLOW SENSOR Node MCU**

Gnd (black) Gnd

Vcc (red) 3V3

DATA pin(yellow) D2

**SERVO MOTOR Node MCU**

Gnd(brown) Gnd

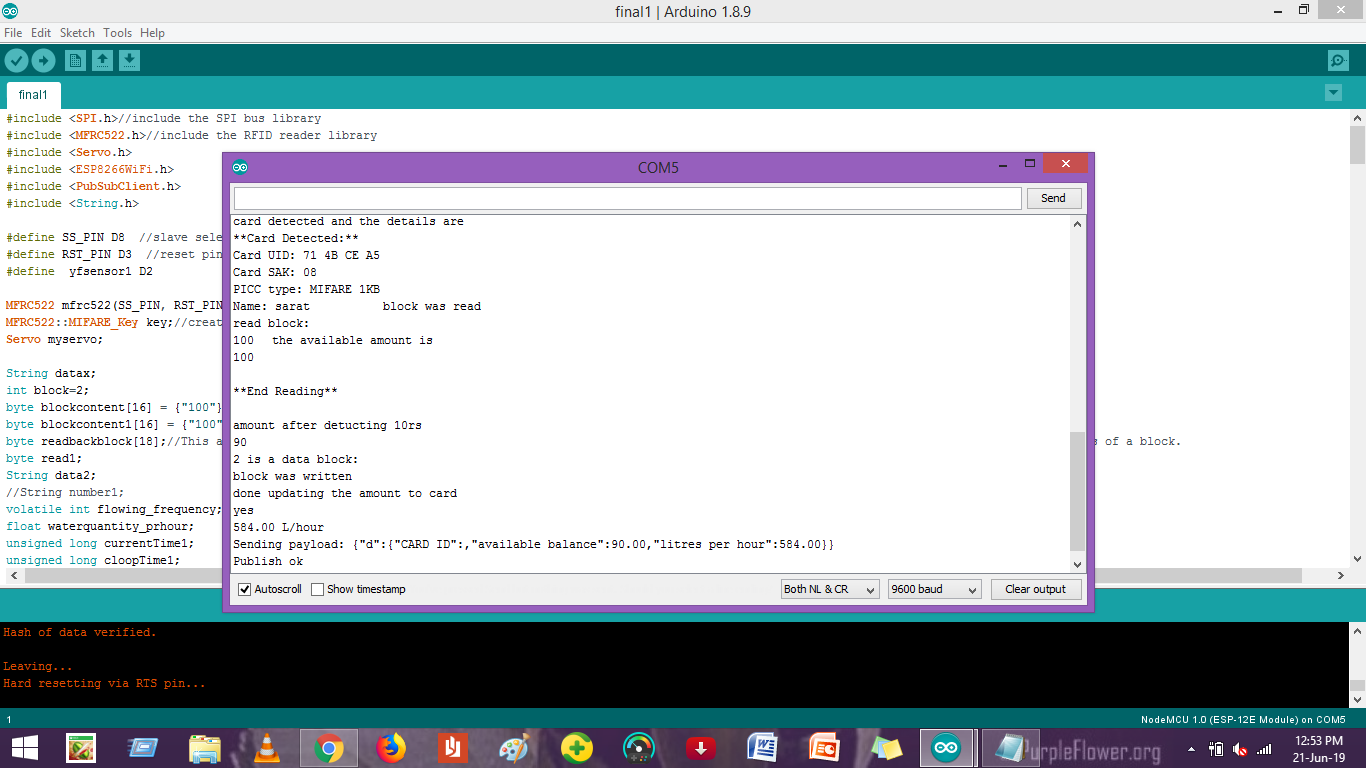
Vcc(red) 3V3

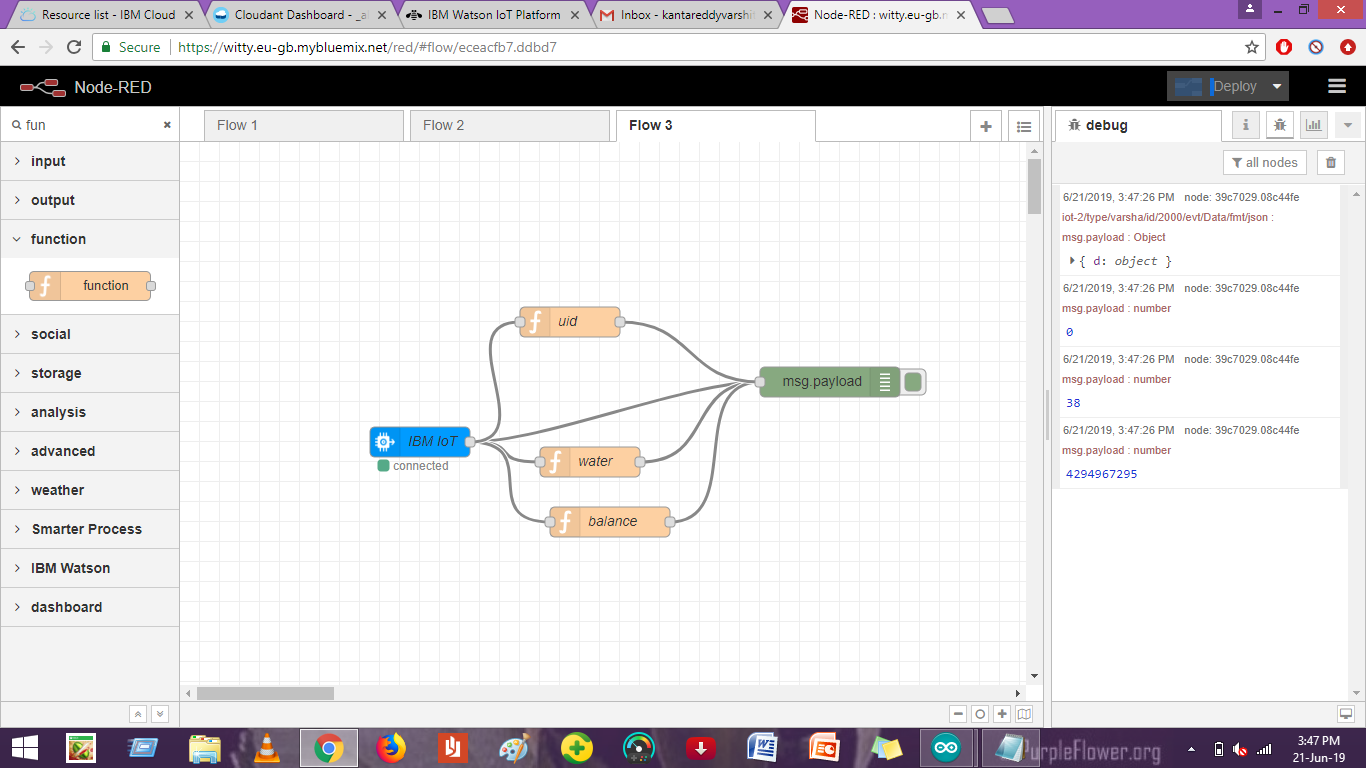
DATA pin(orange) D2

* After the connections are made, upload the code and open the serial monitor

Make sure that the baud rate should be same in code and serial monitor and the connections shouldn’t be loose.

Check The output in serial monitor



* Connect the node-Red flow to the code.
* Define the respective nodes, deploy and check the output.
* **HARDWARES:**
* NodeMCU ESP8266
* RFID reader and cards

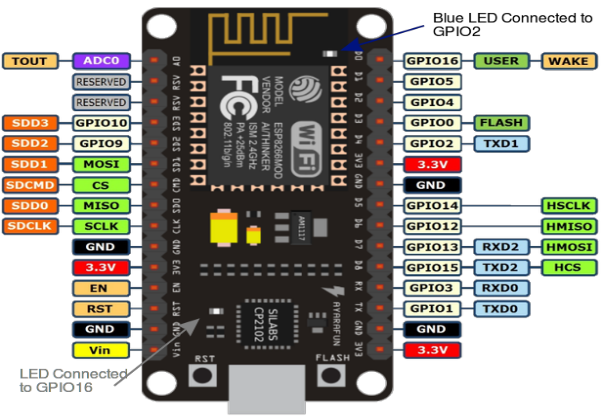
NodeMCU:



**Developer:** ESP8266OpensourceCommunity  
**Type:**  Single-boardmicrocontroller  
**Operatingsystem:** XTOS  
**CPU:** ESP8266  
**Memory:** 128kBytes  
**Storage:** 4MBytes  
**PowerBy:** USB  
**Power Voltage :** 3v ,5v (used with 3.3v Regulator which inbuilt on Board using Pin VIN)  
**Code:** ArduinCp

**IDEUsed:** ArduinoIDE

**GPIO :** 10

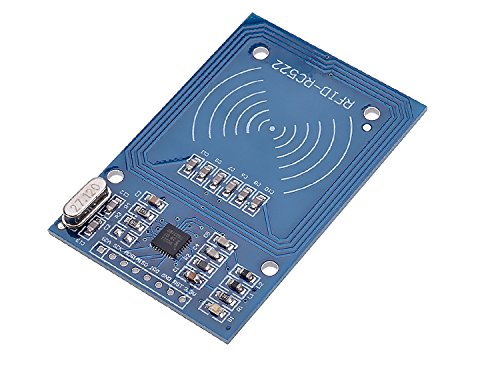


**RFID TECHNOLOGY:**

1.RFID READER:

SPECIFICATIONS:

* Operating Current :13-26mA / DC 3.3V
* Idle Current :10-13mA / DC 3.3V
* Sleep Current: < 80uA
* Peak Current: < 30mA
* Operating Frequency: 13.56MHz
* Supported card types: mifare1 S50, mifare1 S70 MIFARE Ultralight, mifare Pro, MIFARE DESFire
* Environmental Operating Temperature: -20 - 80 degrees Celsius
* Environmental Storage Temperature: -40 - 85 degrees Celsius
* Relative humidity: relative humidity 5% - 95%
* Reader Distance: ≥ 50mm / 1.95" (mifare 1)
* Module Size: 40mm × 60mm
* Module interface: SPI
* Data transfer rate: Maximum 10Mbit/s



2.RFID CARD:

SPECIFICATIONS:

* MFRC522 chip based board
* Operating frequency: 13.56MHz
* Supply Voltage: 3.3V
* Current: 13-26mA
* Read Range: Approx 3cm with supplied card and fob
* SPI Interface
* Max Data Transfer Rate: 10Mbit / s
* Dimensions: 60mm × 39mm

Servomotor:

Working principle:

**Servo motor** works on the PWM ( Pulse Width Modulation ) **principle**, which means its angle of rotation is controlled by the duration of pulse applied to its control PIN.

Specifications:

**Size**

32 × 11.5 × 24mm

**Weight**

9.3g (Include a cable and a connector)

**Speed**

0.12sec/60degrees (4.8V)

**Torque**

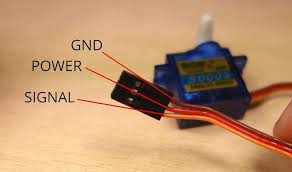
1.5kgf-cm (4.8V

**Voltage**

4.8V-6.0V

**Connector type**

JR type (Yellow: Signal, Red: VCC, Brown: GND)



FLOW SENSOR:

Working principle:

**Flow** is proportional to the amount of deflection. Magnetic **flow sensors** operate on the **principle** that the movement of a conductor through a magnetic field induces a voltage. They consist of a **flow** tube which generates the magnetic field in the pipeline, and an electronic converter that measures induced voltage.

**Specifications**:

* Working voltage 5V-24V
* Maximum current 15 mA
* Weight 43 g
* External diameters 20mm
* Flow rate range 1～30 L/min
* Operating temperature 0℃～80℃



FINAL CODE:

#include <SPI.h>//include the SPI bus library

#include <MFRC522.h>//include the RFID reader library

#include <Servo.h>

#include <ESP8266WiFi.h>

#include <PubSubClient.h>

#include <String.h>

#define SS\_PIN D8 //slave select pin

#define RST\_PIN D3 //reset pin

#define yfsensor1 D2

MFRC522 mfrc522(SS\_PIN, RST\_PIN); // instatiate a MFRC522 reader object.

MFRC522::MIFARE\_Key key;//create a MIFARE\_Key struct named 'key', which will hold the card information

Servo myservo;

unsigned long uid;

String datax;

int block=2;

byte blockcontent[16] = {"100"};

byte blockcontent1[16] = {"100"};

byte readbackblock[18];//This array is used for reading out a block. The MIFARE\_Read method requires a buffer that is at least 18 bytes to hold the 16 bytes of a block.

byte read1;

String data2;

//String number1;

volatile int flowing\_frequency;

float waterquantity\_prhour;

unsigned long currentTime1;

unsigned long cloopTime1;

int duration;

int level;

const char\* ssid = "MANAS";

const char\* password = "tipsy351";

#define ORG "hddkmq"

#define DEVICE\_TYPE "varsha"

#define DEVICE\_ID "2000"

#define TOKEN "8074952339"

//-------- Customise the above values --------

char server[] = ORG ".messaging.internetofthings.ibmcloud.com";

char topic[] = "iot-2/evt/Data/fmt/json";

char authMethod[] = "use-token-auth";

char token[] = TOKEN;

char clientId[] = "d:" ORG ":" DEVICE\_TYPE ":" DEVICE\_ID;

WiFiClient wifiClient;

PubSubClient client(server, 1883,wifiClient);

ICACHE\_RAM\_ATTR void rpm()

{

flowing\_frequency++;

}

void setup() {

Serial.begin(9600);

Serial.println();

//String number1=dump\_byte\_array(mfrc522.uid.uidByte, mfrc522.uid.size);

Serial.print("Connecting to ");

Serial.print(ssid);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.print("WiFi connected, IP address: ");

Serial.println(WiFi.localIP());

SPI.begin(); // Init SPI bus

mfrc522.PCD\_Init(); // Init MFRC522 card (in case you wonder what PCD means: proximity coupling device)

Serial.println("Scan a card");

for (byte i = 0; i < 6; i++) {

key.keyByte[i] = 0xFF;//keyByte is defined in the "MIFARE\_Key" 'struct' definition in the .h file of the library

}

myservo.attach(D1);

pinMode(yfsensor1,INPUT);

}

void loop() {

if ( ! mfrc522.PICC\_IsNewCardPresent()) {//if PICC\_IsNewCardPresent returns 1, a new card has been found and we continue

return;//if it did not find a new card is returns a '0' and we return to the start of the loop

uid = getID();

if(uid != -1){

Serial.print("Card detected, UID: "); Serial.println(uid);

}

}

if ( ! mfrc522.PICC\_ReadCardSerial()) {//if PICC\_ReadCardSerial returns 1, the "uid" struct (see MFRC522.h lines 238-45)) contains the ID of the read card.

return;//if it returns a '0' something went wrong and we return to the start of the loop

}

/\*Serial.print(F("Card UID:"));

dump\_byte\_array(mfrc522.uid.uidByte, mfrc522.uid.size);

Serial.println();\*/

Serial.println("card detected and the details are");

uid = getID();

if(uid != -1){

Serial.print("Card detected, UID: "); Serial.println(uid);}

namedata();

readBlock(block, readbackblock);//read the block back

Serial.println("read block: ");

for (int j=0 ; j<6 ; j++)//print the block contents

{

Serial.write (readbackblock[j]);

}

read1 = (((readbackblock[0]-'0')\*100) + ((readbackblock[1]-'0')\*10) +((readbackblock[2]-'0')\*1));

Serial.println("the available amount is");

Serial.println(read1);

Serial.println("\n\*\*End Reading\*\*\n");

//writeBlock(block,blockcontent1);

amountDeduction();

mfrc522.PICC\_HaltA();

mfrc522.PCD\_StopCrypto1();

float r=read1;

delay(5000);

int pos;

for (pos = 0; pos <= 90; pos += 1) { // goes from 0 degrees to 180 degrees

// in steps of 1 degree

myservo.write(pos); // tell servo to go to position in variable 'pos'

delay(15); // waits 15ms for the servo to reach the position

}

attachInterrupt(4, rpm, RISING); // Setup Interrupt m

currentTime1 = millis();

// Every second, calculate and print litres/hour

if(currentTime1 >= (cloopTime1 + 1000))

{ Serial.println("yes");

cloopTime1 = currentTime1; // Updates cloopTime

waterquantity\_prhour = (flowing\_frequency \* 60 / 7.5); // (Pulse frequency x 60 min) / 7.5Q = flow rate in L/hour

flowing\_frequency = 0; // Reset Counter

Serial.print(waterquantity\_prhour); // Print litres/hour

Serial.println(" L/hour");

}

float w=waterquantity\_prhour;

delay(1000);

if(w>=300)

{

for (pos = 90; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees

myservo.write(pos); // tell servo to go to position in variable 'pos'

delay(15); // waits 15ms for the servo to reach the position

}

}

PublishData(w,r,uid);

delay(100);

}

void PublishData(float water, float balance,unsigned long uid){

if (!!!client.connected()) {

Serial.print("Reconnecting client to ");

Serial.println(server);

while (!!!client.connect(clientId, authMethod, token)) {

Serial.print(".");

delay(500);

}

Serial.println();

}

String payload = "{\"d\":{\"balance\":";

///payload+="," "\"balance\":";

payload += balance;

payload+="," "\"uid\":";

payload += uid;

payload+="," "\"water\":";

payload += water;

payload += "}}";

Serial.print("Sending payload: ");

Serial.println(payload);

if (client.publish(topic, (char\*) payload.c\_str())) {

Serial.println("Publish ok");

} else {

Serial.println("Publish failed");

}

}

void amountDeduction()

{

read1=read1-10;

Serial.println("amount after detucting 10rs");

data2=String(read1);

data2.getBytes(blockcontent, 16);

writeByteArray(blockcontent, 16);

writeBlock(block,blockcontent);

Serial.println("done updating the amount to card");

delay(1000);

}

int writeBlock(int blockNumber, byte arrayAddress[])

{

//this makes sure that we only write into data blocks. Every 4th block is a trailer block for the access/security info.

int largestModulo4Number=blockNumber/4\*4;

int trailerBlock=largestModulo4Number+3;//determine trailer block for the sector

if (blockNumber > 2 && (blockNumber+1)%4 == 0){Serial.print(blockNumber);Serial.println(" is a trailer block:");return 2;}//block number is a trailer block (modulo 4); quit and send error code 2

Serial.print(blockNumber);

Serial.println(" is a data block:");

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*authentication of the desired block for access\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

byte status = mfrc522.PCD\_Authenticate(MFRC522::PICC\_CMD\_MF\_AUTH\_KEY\_A, trailerBlock, &key, &(mfrc522.uid));

if (status != MFRC522::STATUS\_OK) {

Serial.print("PCD\_Authenticate() failed: ");

mfrc522.GetStatusCodeName((MFRC522::StatusCode)status);

Serial.println( mfrc522.GetStatusCodeName((MFRC522::StatusCode)status));

return 3;//return "3" as error message

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*writing the block\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

status = mfrc522.MIFARE\_Write(blockNumber, arrayAddress, 16);//valueBlockA is the block number, MIFARE\_Write(block number (0-15), byte array containing 16 values, number of bytes in block (=16))

//status = mfrc522.MIFARE\_Write(9, value1Block, 16);

if (status != MFRC522::STATUS\_OK) {

Serial.print("MIFARE\_Write() failed: ");

mfrc522.GetStatusCodeName((MFRC522::StatusCode)status);

Serial.println( mfrc522.GetStatusCodeName((MFRC522::StatusCode)status));

return 4;//return "4" as error message

}

Serial.println("block was written");

}

int readBlock(int blockNumber, byte arrayAddress[])

{

int largestModulo4Number=blockNumber/4\*4;

int trailerBlock=largestModulo4Number+3;//determine trailer block for the sector

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*authentication of the desired block for access\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

byte status = mfrc522.PCD\_Authenticate(MFRC522::PICC\_CMD\_MF\_AUTH\_KEY\_A, trailerBlock, &key, &(mfrc522.uid));

if (status != MFRC522::STATUS\_OK) {

Serial.print("PCD\_Authenticate() failed (read): ");

mfrc522.GetStatusCodeName((MFRC522::StatusCode)status);

Serial.println( mfrc522.GetStatusCodeName((MFRC522::StatusCode)status));

return 3;//return "3" as error message

}

byte buffersize = 18;//we need to define a variable with the read buffer size, since the MIFARE\_Read method below needs a pointer to the variable that contains the size...

status = mfrc522.MIFARE\_Read(blockNumber, arrayAddress, &buffersize);//&buffersize is a pointer to the buffersize variable; MIFARE\_Read requires a pointer instead of just a number

if (status != MFRC522::STATUS\_OK) {

Serial.print("MIFARE\_read() failed: ");

mfrc522.GetStatusCodeName((MFRC522::StatusCode)status);

Serial.println( mfrc522.GetStatusCodeName((MFRC522::StatusCode)status));

return 4;//return "4" as error message

}

Serial.println("block was read");

}

void namedata(){

// Prepare key - all keys are set to FFFFFFFFFFFFh at chip delivery from the factory.

MFRC522::MIFARE\_Key key;

for (byte i = 0; i < 6; i++) key.keyByte[i] = 0xFF;

//some variables we need

byte block;

byte len;

MFRC522::StatusCode status;

Serial.println(F("\*\*Card Detected:\*\*"));

mfrc522.PICC\_DumpDetailsToSerial(&(mfrc522.uid)); //dump some details about the card

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

byte buffer1[18];

block = 4;

len = 18;

//------------------------------------------- GET FIRST NAME

status = mfrc522.PCD\_Authenticate(MFRC522::PICC\_CMD\_MF\_AUTH\_KEY\_A, 4, &key, &(mfrc522.uid)); //line 834 of MFRC522.cpp file

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;

}

//PRINT FIRST NAME

for (uint8\_t i = 0; i < 16; i++)

{

if (buffer1[i] != 32)

{

Serial.write(buffer1[i]);

}

}

Serial.print(" ");

delay(1000); //change value if you want to read cards faster

}

void writeByteArray(byte array[], int arrlength)

{

for (int j = 0 ; j < arrlength ; j++) //print the block contents

{

blockcontent[j]=array[j];

Serial.write (array[j]);//Serial.write() transmits the ASCII numbers as human readable characters to serial monitor

}

Serial.println("");

}

/\*void dump\_byte\_array(byte \*buffer, byte bufferSize) {

for (byte i = 0; i < bufferSize; i++) {

Serial.print(buffer[i] < 0x10 ? " 0" : " ");

Serial.print(buffer[i], HEX);

}

}\*/

unsigned long getID(){

if ( ! mfrc522.PICC\_ReadCardSerial()) { //Since a PICC placed get Serial and continue

return -1;

}

unsigned long hex\_num;

hex\_num = mfrc522.uid.uidByte[0] << 24;

hex\_num += mfrc522.uid.uidByte[1] << 16;

hex\_num += mfrc522.uid.uidByte[2] << 8;

hex\_num += mfrc522.uid.uidByte[3];

mfrc522.PICC\_HaltA(); // Stop reading

return hex\_num;

}

ADVANTAGES:

* Organised water will be supplied.
* Man power will be low.
* Saves money.
* Scarcity will be reduced.
* The water supply process will be systematic.
* Get rids of calculation errors.
* Get paid on time.
* This process is more securable process for transaction of money.