Vehicle Heavy Weight Monitoring

Accident Prevention System

– Final Code 😀 –

```
#include <Wire.h>

#include <LiquidCrystal_I2C.h>

#include <SPI.h>

#include <MFRC522.h>

#include <HX711.h>

#include <Servo.h>


#define SS_PIN 10

#define RST_PIN 5


MFRC522 mfrc522(SS_PIN, RST_PIN);

MFRC522::MIFARE_Key key;


int block = 2;

byte readbackblock[18];


// Load Cell

const int DOUT_PIN = 2;

const int SCK_PIN = 3;
```

```
long weight;
long actualWeight;



HX711 scale;


// LCD SCREEN
LiquidCrystal_I2C lcd(0x27, 16, 2);
void setup() {
  Serial.begin(9600);
  SPI.begin();
  mfrc522.PCD_Init();


  // Prepare the security key for the read function.
  for (byte i = 0; i < 6; i++) {
    key.keyByte[i] = 0xFF;
  }


  scale.begin(DOUT_PIN, SCK_PIN);
  Serial.println("HX711 Initialized");


  lcd.init();
  lcd.backlight();
  lcd.clear();

  lcd.setCursor(0, 0);

  lcd.print("Welcome!");

}


int extractInteger(byte* data) {

  // Assuming the first two bytes represent the integer value

  int extractedValue = (data[0] - '0') * 10 + (data[1] - '0');

  return extractedValue;

}


void loop() {


  // LOAD CELL


    // Print the LOAD Cell value

    weight = scale.get_units();

    actualWeight = ((fabs(weight - 253500) * 2) - 3000)/1000;

     Serial.print("LOAD CELL : ");

    Serial.println(actualWeight);



  // Look for new cards
```

```
Serial.println("Scan TAG");

if (!mfrc522.PICC_IsNewCardPresent()) {

  delay(2000);

  return;

}


// Select one of the cards

if (!mfrc522.PICC_ReadCardSerial()) {

  return;

}


// Read the specified block from RFID

if (readBlock(block, readbackblock)) {

  // Print the RFID block contents

  Serial.print("Read block from RFID: ");

  for (int j = 0; j < 16; j++) {

    Serial.write(readbackblock[j]);

  }

  Serial.println();


  // Extract an integer from the block data

  int extractedValue = extractInteger(readbackblock);
```

```
// Print the extracted value
Serial.print("Extracted Integer: ");
Serial.println(extractedValue);

if (actualWeight <= extractedValue) {
  // Open barriers
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Gate Opened");
  // myservo.write(90);
  delay(1500);
  // myservo.write(0);
  // lcd.clear();
} else {
  // Display a message indicating vehicle overload
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Vehicle Overload");
  lcd.setCursor(0, 1);
  lcd.print("Not Allowed");
  delay(1500);
  // lcd.clear();
}
```

```
  }


  // Halt the card and stop encryption

  mfrc522.PICC_HaltA();

  mfrc522.PCD_StopCrypto1();


  // Delay before looking for a new card

  delay(5000);

}


// Read specific block

bool readBlock(int blockNumber, byte arrayAddress[]) {

  int trailerBlock = (blockNumber / 4) * 4 + 3;


  // Authentication of the desired block for access

  if (mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A,
trailerBlock, &key, &(mfrc522.uid)) != MFRC522::STATUS_OK) {

    Serial.println("Authentication failed");

    return false;

  }


  // Reading a block

  byte bufferSize = 18;
  if (mfrc522.MIFARE_Read(blockNumber, arrayAddress, &bufferSize) !=
MFRC522::STATUS_OK) {

    Serial.println("Read failed");

    return false;

  }



  // Print status

  Serial.println("Block read successful");

  return true;

}
```

Rfid :-

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

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

```
#define SS_PIN 10  //slave select pin

#define RST_PIN 5  //reset pin


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


//this is the block number we will write into and then read.

int block=2;


byte blockcontent[16] = {"20tons"};  //an array with 16 bytes to be
written into one of the 64 card blocks is defined

//byte blockcontent[16] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};  //all zeros.
This can be used to delete a block.


//This array is used for reading out a block.

byte readbackblock[18];


void setup()

{

    Serial.begin(9600);        // Initialize serial communications with
the PC

    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 MIFARE Classic card");


  // Prepare the security key for the read and write functions.
  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
  }
}


void loop()
{
  // Look for new cards
  if ( ! mfrc522.PICC_IsNewCardPresent()) {
    return;
  }


  // Select one of the cards
  if ( ! mfrc522.PICC_ReadCardSerial())
  {
    return;
  }
    Serial.println("card selected");
```

```
   //the blockcontent array is written into the card block

  //  writeBlock(block, blockcontent);


   //read the block back

   readBlock(block, readbackblock);

   //uncomment below line if you want to see the entire 1k memory with the
block written into it.

   //mfrc522.PICC_DumpToSerial(&(mfrc522.uid));


   //print the block contents

   Serial.print("read block: ");

   for (int j=0 ; j<16 ; j++)

   {

     Serial.write (readbackblock[j]);

   }

   Serial.println("");

}





//Write specific block

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;}

  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: ");

         Serial.println(mfrc522.GetStatusCodeName(status));

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

  }



  //writing the block

  status = mfrc522.MIFARE_Write(blockNumber, arrayAddress, 16);

  //status = mfrc522.MIFARE_Write(9, value1Block, 16);

  if (status != MFRC522::STATUS_OK) {

           Serial.print("MIFARE_Write() failed: ");
```

```
          Serial.println(mfrc522.GetStatusCodeName(status));

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

  }

  Serial.println("block was written");

}




//Read specific block

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): ");

         Serial.println(mfrc522.GetStatusCodeName(status));

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

  }
```

```
//reading a block
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: ");
          Serial.println(mfrc522.GetStatusCodeName(status));
          return 4;//return "4" as error message
  }
  Serial.println("block was read");
}



// #include <SPI.h>
// #include <MFCE522.h>
// #include <Servo.h>

// #define SS_PIN 10
// #define PST_PIN 9
// #define LED_G 4
// #define LED_R 5
// #define BUZZER 2

// MFRC522 mfrc522(SS_PIN,RST_PIN);

// Servo myServo;


// void setup() {

//   // put your setup code here, to run once:

//   Serial.begin(9600);   // Serial Comm

//   SPI.begin();          // SPI BUS

//   mfrc522.PDC_Init();

//   myServo.attach(3);

//   myServo.write(0);     // Start Position

//   pinMode(LED_G,OUTPUT);

//   pinMode(LED_R,OUTPUT);

//   pinMode(BUZZER,OUTPUT);

//   noTone(BUZZER);

//   Serial.println("Put Your CArd To THE READER");

// }


// void loop() {

//   // put your main code here, to run repeatedly:



// }
```

Load Cell :-

```
#include "HX711.h"


// Pin configuration

const int DOUT_PIN = 2;

const int SCK_PIN = 3;


HX711 scale;


void setup() {

  Serial.begin(9600);

  scale.begin(DOUT_PIN, SCK_PIN);

}


void loop() {

  if (scale.is_ready()) {

    // long weight = 53400+scale.get_units();  // Get the weight in grams

    // Serial.print((fabs(weight)*2)-3000);

    // long weight = scale.get_units();  // Get the weight in grams

    // Serial.print((fabs(weight-253500)*2)-5000);
    long weight = scale.get_units();  // Get the weight in grams

    Serial.print("Weight: ");

    Serial.print((fabs(weight-253500)*2)-3000);

    Serial.println(" grams");

  } else {

    Serial.println("Error reading from the scale.");

  }



  delay(1000);  // Delay for stability

}
```

LCD :-

```
/*

   Arduino LCD I2C Tutorial by 'Tronics Lk"

   Please visit my YouTube channel from this link for more Tutorials


https://www.youtube.com/channel/UCYJa3gs8q49-N3TLm-7ygUw?sub_confirmation=
1

*/

#include <Wire.h>
```

```
#include <LiquidCrystal_I2C.h>


LiquidCrystal_I2C lcd(0x27, 16, 2); // set the LCD address to 0x27 for a
16 chars and 2 line display


void setup()

{

  lcd.init();          // initialize the lcd

  lcd.backlight();     // Turn on the LCD screen backlight

}


void loop()

{

  lcd.setCursor(4, 0);

  lcd.print("Hello World.!");

  delay(3000);

  lcd.clear();


  lcd.setCursor(0, 1);

  lcd.print("This is a LCD ");

  lcd.setCursor(2, 1);

  lcd.print("Screen Test");

  delay(3000);
  lcd.clear();

}
```

Servo Motor :-

```
#include <Servo.h>


Servo myservo;  // create servo object to control a servo


void setup() {

  myservo.attach(9);  // attaches the servo on pin 9 to the servo object

}


void loop() {

  myservo.write(90);
  delay(200);                                  // wait for 2 seconds

  myservo.write(0);

  delay(1000);                                  // wait for 2 seconds

  myservo.write(90);

  delay(1000);                                  // wait for 2 seconds

  myservo.write(0);

  delay(1000);                                  // wait for 2 seconds

  myservo.write(90);

  delay(1000);                                  // wait for 2 seconds

}
```