Computer Interface Course Project Report
4th Year Computer Engineering

Project Title: [RFID Attendance System]

Team ID: [B1]

Team Members:

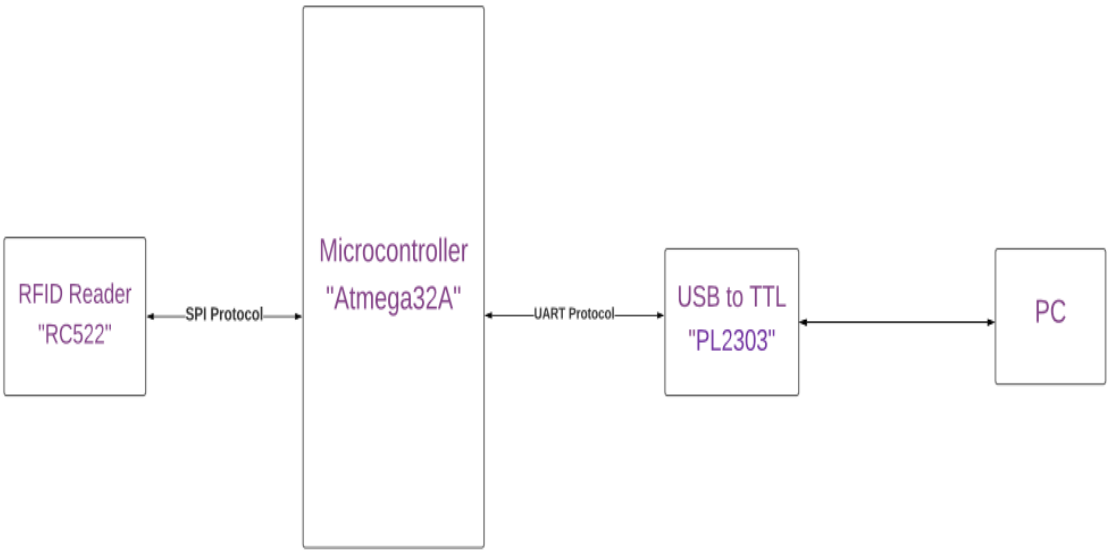
SN	Student Name	Section Number
1	Rahma Mohamed Makram	2
2	Rana Mohamed Hussein Mohamed Bekheet	2
3	Sarah Gamal El-Deen Mohamed	2
4	Solwan Shokry Ahmed Mohamed	2
5	-	-

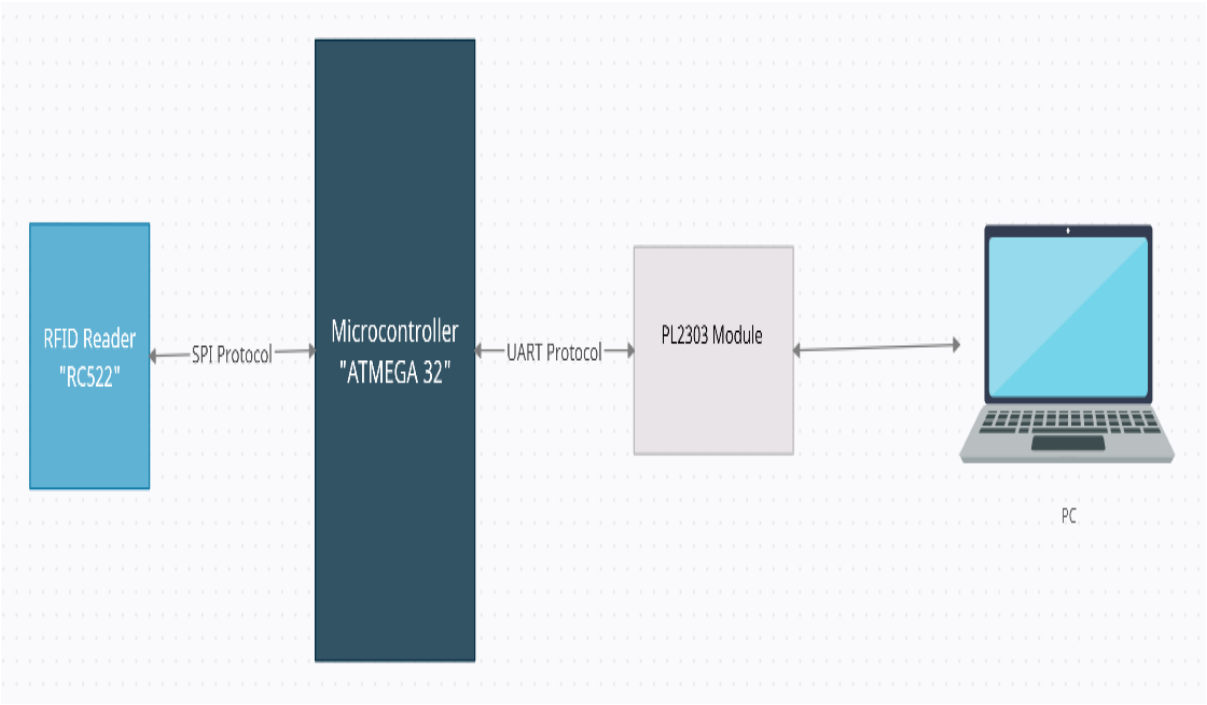
1. Project Objective:

Attendance is an action of particular person being present on event at work or an educational institution, Taking attendance manually consumes more time, effort and error therefore an automated attendance system can be very helpful. Our System allows the user that carries an RFID Tag which has a distinct ID to register his/her ID in the system, later on the user can scan his/her RFID tag ,the RFID reader reads the ID ,the ID is then extracted from the database with his full information like, name, his picture marking the attendance of the person carrying the tag

2. System Block Diagram:

2.1 Block Diagram:

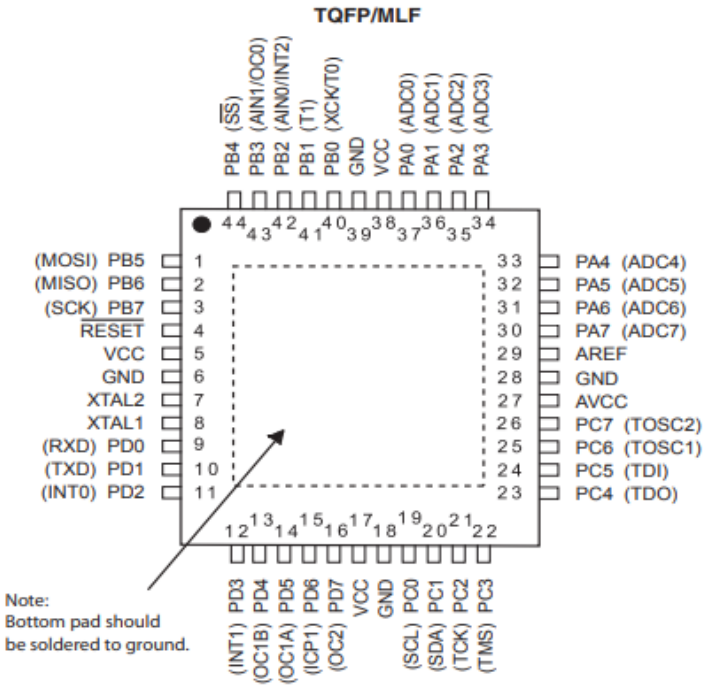


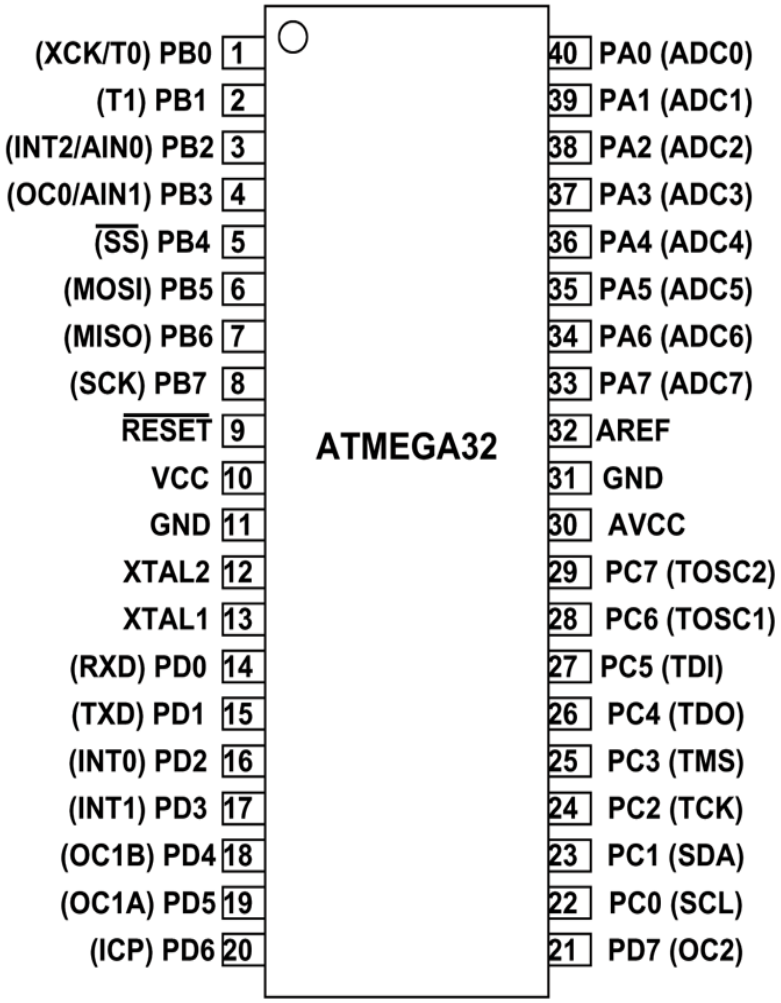


2.2 Block Diagram Description:

1- Atmega32A:

The ATmega32A is a low power, CMOS 8-bit microcontrollers based on the AVR® enhanced RISC architecture, The ATmega32A is a 40/44-pins device with 32 KB Flash, 2 KB SRAM and 1 KB EEPROM. By executing instructions in a single clock cycle, the devices achieve CPU throughput approaching one million instructions per second (MIPS) per megahertz, allowing the system designer to optimize power consumption versus processing speed. The ATmega32A provides two way in the system





Pin Descriptions:

VCC: Digital supply voltage

GND :Ground.

Port A (PA7:PA0): Port A serves as the analog inputs to the A/D Converter. Port A also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used. Port pins can provide internal pull-up resistors (selected for each bit). The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. When pins PA0 to PA7 are used as inputs and are externally pulled low, they will source current if the internal pull-up resistors are activated. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B (PB7:PB0): Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are

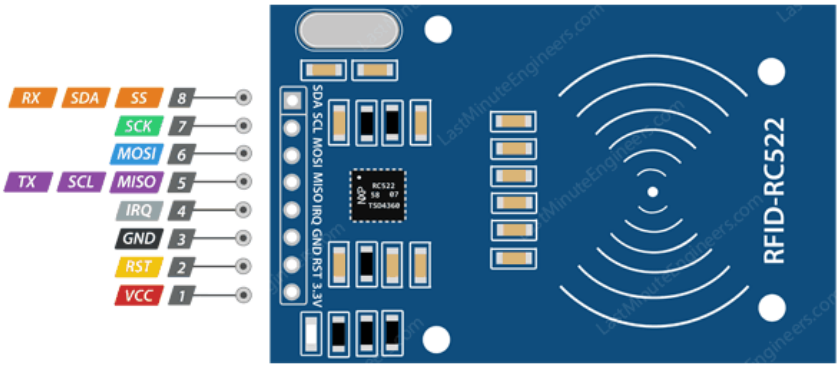
activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.
Port C (PC7:PC0): Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running. If the JTAG interface is enabled, the pull-up resistors on pins PC5(TDI), PC3(TMS) and PC2(TCK) will be activated even if a reset occurs. The TD0 pin is tri-stated unless TAP states that shift out data are entered. Port C also serves the functions of the JTAG interface.
Port D (PD7:PD0): Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.
RESET : Reset Input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. Shorter pulses are not ensured to generate a reset.
XTAL1 : Input to the inverting Oscillator amplifier and input to the internal clock operating circuit. 2.2.9 XTAL2 Output from the inverting Oscillator amplifier.
AVCC: AVCC is the supply voltage pin for Port A and the A/D Converter. It should be externally connected to VCC, even if the ADC is not used. If the ADC is used, it should be connected to VCC through a low-pass filter.
AREF : AREF is the analog reference pin for the A/D Converter

2- RFID Reader RC522:

RFID or Radio Frequency Identification system consists of two main components, a transponder/tag attached to an object to be identified, and a Transceiver also known as interrogator/Reader. A Reader consists of a Radio Frequency module and an antenna which generates high frequency electromagnetic field. On the other hand, the tag is usually a passive device, meaning it doesn’t contain a battery. Instead it contains a microchip that stores and processes information, and an antenna to receive and transmit a signal.

To read the information encoded on a tag, it is placed in close proximity to the Reader (does not need to be within direct line-of-sight of the reader). A Reader generates an electromagnetic field which causes electrons to move through the tag's antenna and subsequently power the chip.

The powered chip inside the tag then responds by sending its stored information back to the reader in the form of another radio signal. This is called backscatter. The backscatter, or change in the electromagnetic/RF wave, is detected and interpreted by the reader which then sends the data out to a computer or microcontroller.



VCC supplies power for the module. This can be anywhere from 2.5 to 3.3 volts. You can connect it to 3.3V output from your Arduino. Remember connecting it to 5V pin will likely destroy your module!

RST is an input for Reset and power-down. When this pin goes low, hard power-down is enabled. This turns off all internal current sinks including the oscillator and the input pins are disconnected from the outside world. On the rising edge, the module is reset.

GND is the Ground Pin and needs to be connected to GND pin on the Arduino.

IRQ is an interrupt pin that can alert the microcontroller when RFID tag comes into its vicinity.

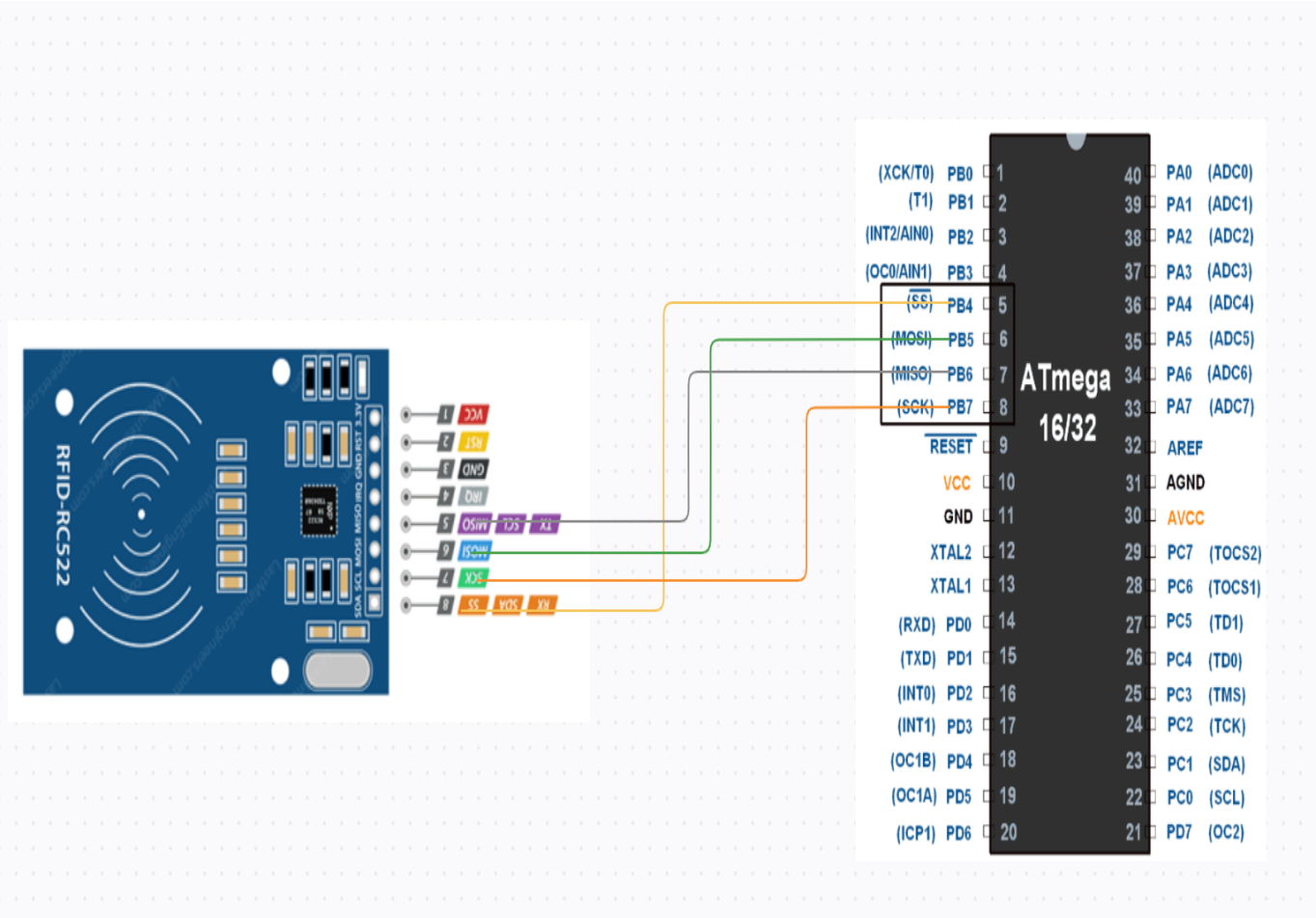
MISO / SCL / Tx pin acts as Master-In-Slave-Out when SPI interface is enabled, acts as serial clock when I2C interface is enabled and acts as serial data output when UART interface is enabled.

MOSI (Master Out Slave In) is SPI input to the RC522 module.

SCK (Serial Clock) accepts clock pulses provided by the SPI bus Master i.e. Arduino.

SS / SDA / Rx pin acts as Signal input when SPI interface is enabled, acts as serial data when I2C interface is enabled and acts as serial data input when UART interface is enabled. This pin is usually marked by encasing the pin in a square so it can be used as a reference for identifying the other pins.

RC522 Interfacing with Atmega32:



2- PL2303 Module USB to TTL:

Serial communication means transferring a single bit at a time. We can connect a mouse, a modem, a printer, a plotter, another PC, dongles, etc. But its usage (both software and hardware) is a secret to users. But it is not difficult to understand how to connect devices to it and how to program it.

RS-232 used for data exchange between the devices. It specifies common voltage and signal level, common pin wire configuration and minimum, amount of control signals. As mentioned above this standard was designed with specification for electromagnetically teletypewriter and modem system

But with the new PC and laptops, there is no RS232 protocol and DB9 connector. We have to use serial to the USB connector. There are various serial to USB connectors available e.g. CP2102, FT232RL, CH340, etc. PL2303 Module USB to TTL, it's a small USB to TTL serial tool, using the PL2303 chip. You can use it to connect some serial device to your PC via USB port.

Specification

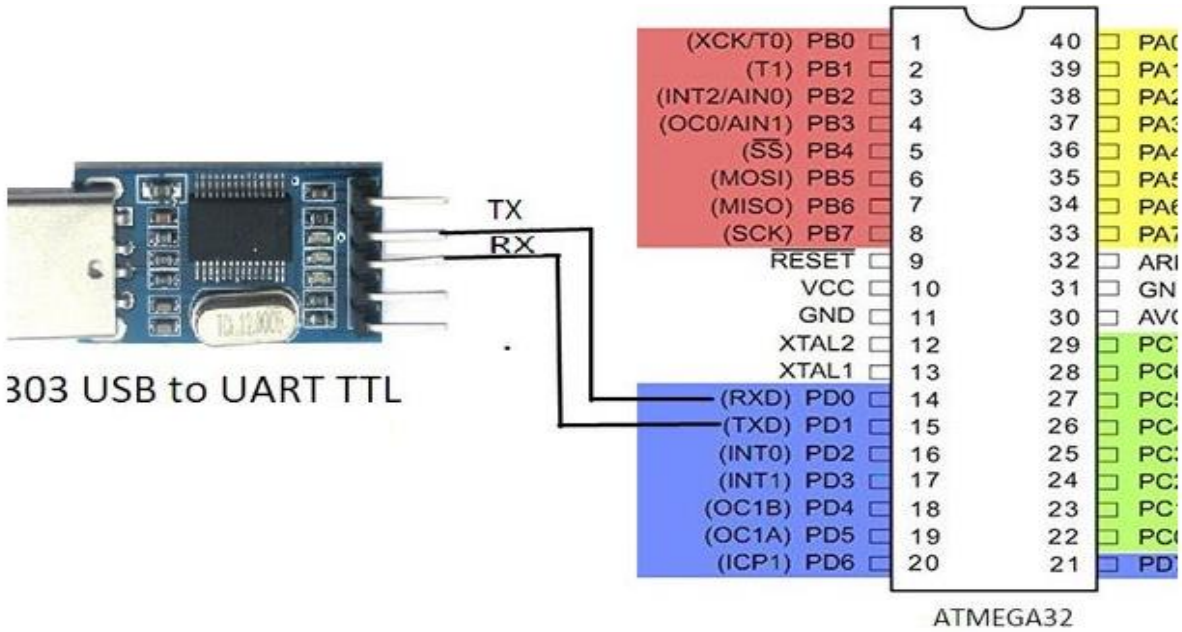
Module Type : AdapterBoard

Size : 4.6 x 1.5 x 1.1cm

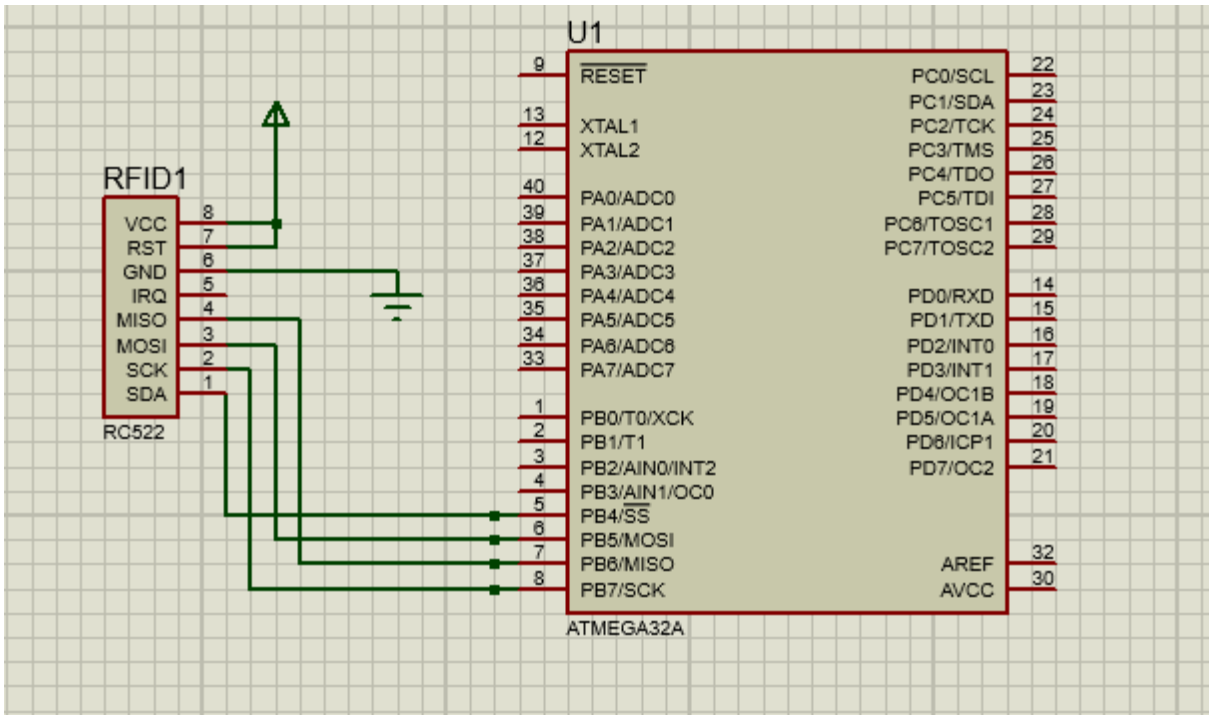
Operation Level : Digital 5V

Power Supply : External 5V

PL2303 Interfacing with Atmega32:



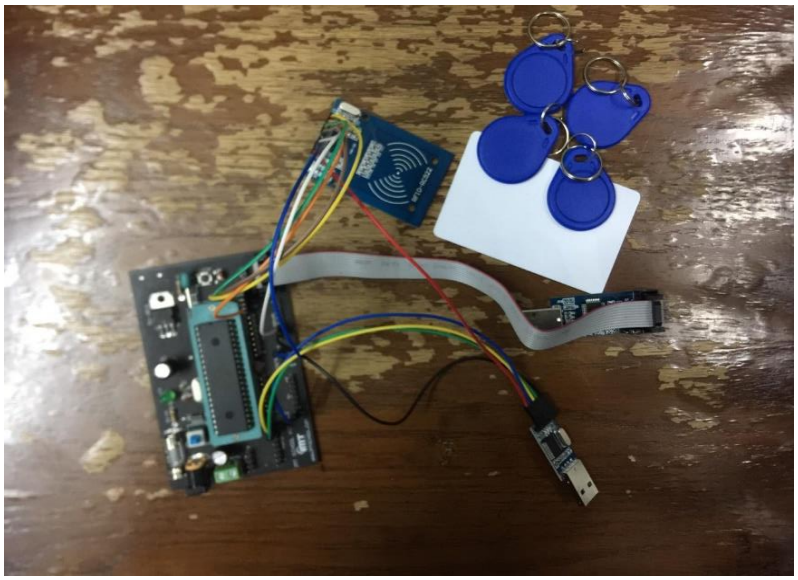
3. Schematic Diagram (Circuit Diagram):

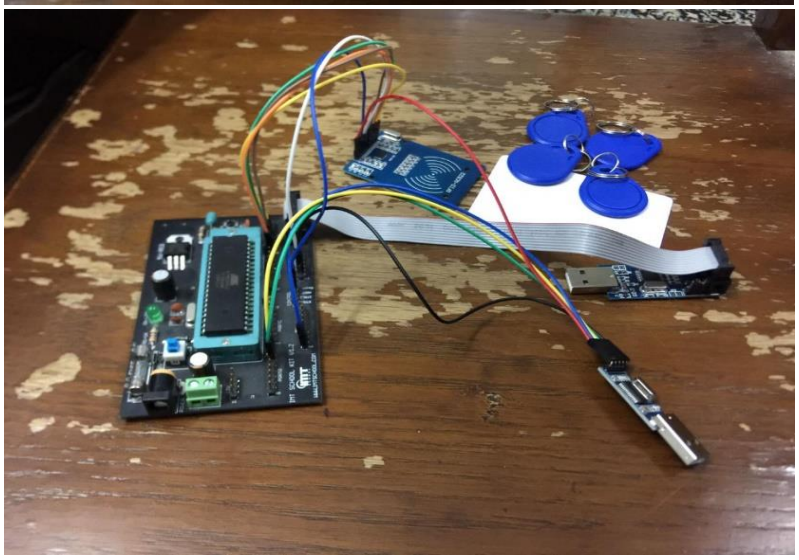


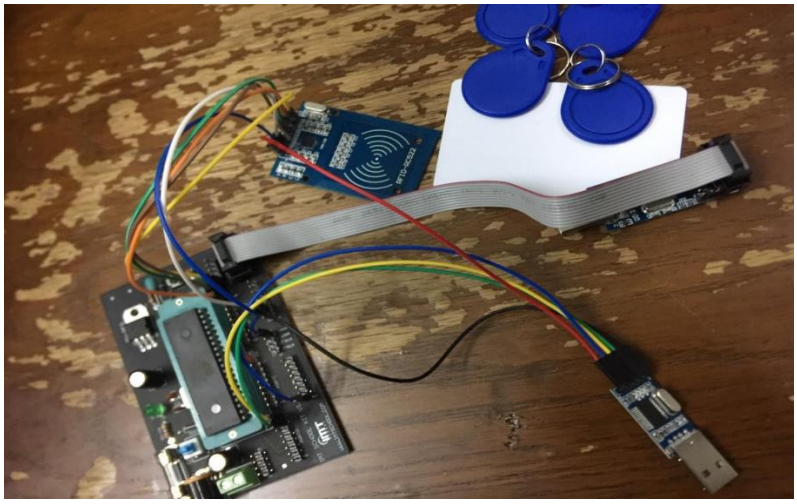
4. List Of Components:

SN	Item Type	Item Code Name	Purpose	Quantity
1	Microcontroller	Atmega32	Receives the ID from the RFID reader and sends it to the database to get the full information related to that ID	1
2	RFID	RC522	Reads the RFID card	1
3	RFID Card	RFID Card	Each card carries a distinct ID	4
4	USB to RS232	PL2303 Module USB to TTL	Provides serial interface to the GUI	1
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

5. Real-Time Hardware Photo:







6. Source Code:

6.1 Hardware-side source code:

1. Embedded-C Source code

```
/*
 * main.c
 */

#include <avr/io.h>
#include <util/delay.h>
#include "utils.h"
#include "spi.h"
#include "mfrc522.h"
#include "uart.h"
uint16 readSuccess;
uint8 defcard[][4] = { {0xC6,0xFD,0xC5,0x32},{0x32,0xD7,0x0F,0xB} };
uint8 N = 4;                        //Variable to store the number of RFID cards/tags we will use
uint8 readcard[4];   //stores the UID of current tag which is read
uint8 version;
uint8 req_mode;  //initialize
uint8 str[MAX_LEN];
uint8 ID[32]="";
//uint8_t reg;
unsigned char getid();
```

```c
unsigned char SelfTestBuffer[64];
void array_to_string(unsigned char *array, unsigned char len,char *buffer);
unsigned char numbers[4]={'1','2','3','4'};
unsigned char string[]="";
int main()
{
  unsigned char byte;

  _delay_ms(50);

  //LCDWriteStringXY(2,0,"RFID Reader");
  //LCDWriteStringXY(5,1,VERSION_STR);


  array_to_string(numbers,4,string);
  UART_init();
  //UART_sendByte('s');
  spi_init();
  _delay_ms(1000);

  //init reader
  mfrc522_init();

  //check version of the reader
  byte = mfrc522_read(VersionReg);
  if(byte == 0x92)
  {
      UART_sendString("MIFARE RC522v2");
      UART_sendString("Detected");
  }
  else if(byte == 0x91 || byte==0x90)
  {
      UART_sendString("MIFARE RC522v1");
      UART_sendString("Detected");
  }
  else
  {
      UART_sendString("No reader found");
  }

  byte = mfrc522_read(ComIEnReg);
```

```
68   mfrc522_write(ComIEnReg,byte|0x20);
69   byte = mfrc522_read(DivIEnReg);
70   mfrc522_write(DivIEnReg,byte|0x80);
71
72   _delay_ms(1500);
73
74
75   while(1){
76
77        byte = getid();
78        _delay_ms(1000);
79        if(byte == CARD_FOUND)
80        {
81             UART_sendString(ID);
82        }
83   }
84 }
85 unsigned char getid()
86 {
87   unsigned char status;
88   status = mfrc522_request(PICC_REQALL, str);
89   if (status == ERROR)
90   {
91        return 0;
92   }
93   if (!((SPSR & (1 << SPIF))))
94   {
95        return 0;
96   }
97
98
99
100       for (int i = 0; i < 4; i++)
101       {
102            readcard[i] = spi_recieve(); //storing the
   UID of the tag in readcard
103
104       }
105       array_to_string(readcard,N,ID);
106       UART_sendString(ID);
107       mfrc522_reset();
108
```

```
109         return 1;
110  }
111  void array_to_string(unsigned char *array, unsigned
   char len,char *buffer)
112  {
113      for (unsigned int i = 0; i < len; i++)
114      {
115       unsigned char nib1 = (array[i] >> 4) & 0x0F;
116       unsigned char nib2 = (array[i] >> 0) & 0x0F;
117          buffer[i*2+0] = nib1  < 0xA ? '0' + nib1  :
   'A' + nib1  - 0xA;
118          buffer[i*2+1] = nib2  < 0xA ? '0' + nib2  :
   'A' + nib2  - 0xA;
119      }
120      buffer[len*2] = '\0';
121  }
122  /*
123   * spi_config.h
124   */
125
126
127  #ifndef SPI_CONFIG_H_
128  #define SPI_CONFIG_H_
129
130  #include <avr/io.h>
131  /*
132   * Set to 1, SPI will work in master mode
133   * else in slave mode
134   */
135  #define SPI_CONFIG_AS_MASTER     1
136
137
138  /*
139   * Config SPI pin diagram
140   */
141  #define SPI_DDR                     DDRB
142  #define SPI_PORT                PORTB
143  #define SPI_PIN                     PINB
144  #define SPI_MOSI                PB5
145  #define SPI_MISO                PB6
146  #define SPI_SS                  PB4
147  #define SPI_SCK                     PB7
```

```c
#endif /* SPI_CONFIG_H_ */
/*
 * spi.h
 */


#ifndef SPI_H_
#define SPI_H_

#include "spi_config.h"
#include <stdint.h>

void spi_init();
uint8_t spi_transmit(uint8_t data);
unsigned char spi_recieve();

#define ENABLE_CHIP() (SPI_PORT &= (~(1<<SPI_SS)))
#define DISABLE_CHIP() (SPI_PORT |= (1<<SPI_SS))




#endif /* SPI_H_ */
/*
 * spi.c
 */
#include "spi.h"

#if SPI_CONFIG_AS_MASTER

void spi_init()
{
	SPI_DDR = (1<<SPI_MOSI)|(1<<SPI_SCK)|(1<<SPI_SS);
	SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0);
//pre-scaler 16
}


```

```c
uint8_t spi_transmit(uint8_t data)
{
    SPDR = data;
    while(!(SPSR & (1<<SPIF)));
    return SPDR;
}
unsigned char spi_recieve()
{
    while(!(SPSR & (1<<SPIF)));
    return SPDR;
}
#else
void spi_init()
{
    SPI_DDR = (1<<SPI_MISO);
    SPCR = (1<<SPE);
}

uint8_t spi_transmit(uint8_t data)
{
    while(!(SPSR & (1<<SPIF)));
    return SPDR;
}



#endif
/*****************************************************************************
 *
 * Module: UART
 *
 * File Name: uart.h
 *
 * Description: Header file for the UART AVR driver
 *
 *****************************************************************************/

#ifndef UART_H_
#define UART_H_
```

```
#include "micro_config.h"
#include "std_types.h"
#include "common_macros.h"

/*****************************************************************************
 *                                Preprocessor Macros                        *
 *****************************************************************************/
#ifndef F_CPU
#define F_CPU 8000000UL //1MHz Clock frequency
#endif


/* UART Driver Baud Rate */
#define USART_BAUDRATE 9600

/*****************************************************************************
 *                                Functions Prototypes                       *
 *****************************************************************************/
void UART_init(void);

void UART_sendByte(const uint8 data);

uint8 UART_recieveByte(void);

void UART_sendString(const uint8 *Str);

void UART_receiveString(uint8 *Str); // Receive until #

#endif /* UART_H_ */
/*****************************************************************************
 *
```

```
 * Module: UART
 *
 * File Name: uart.c
 *
 * Description: Source file for the UART AVR driver
 *
 ********************************************************************************/

#include "uart.h"

#define BAUD_PRESCALE (((F_CPU / (USART_BAUDRATE * 8UL))) - 1)

/*******************************************************************************
 *                      Functions Definitions                                  *
 *******************************************************************************/
void UART_init(void)
{
    /* U2X = 1 for double transmission speed */
    UCSRA = (1<<U2X);
    /************************** UCSRB Description **************************
     * RXCIE = 0 Disable USART RX Complete Interrupt Enable
     * TXCIE = 0 Disable USART Tx Complete Interrupt Enable
     * UDRIE = 0 Disable USART Data Register Empty Interrupt Enable
     * RXEN  = 1 Receiver Enable
     * RXEN  = 1 Transmitter Enable
     * UCSZ2 = 0 For 8-bit data mode
     * RXB8 & TXB8 not used for 8-bit data mode
     ***********************************************************************/
    UCSRB = (1<<RXEN) | (1<<TXEN);
```

```
288
289       /************************* UCSRC Description *************************
290        * URSEL   = 1 The URSEL must be one when writing the UCSRC
291        * UMSEL   = 0 Asynchronous Operation
292        * UPM1:0  = 00 Disable parity bit
293        * USBS    = 0 One stop bit
294        * UCSZ1:0 = 11 For 8-bit data mode
295        * UCPOL   = 0 Used with the Synchronous operation only
296 ********************************************************************************/
297       UCSRC = (1<<URSEL) | (1<<UCSZ0) | (1<<UCSZ1);
298
299       /* First 8 bits from the BAUD_PRESCALE inside UBRRL and last 4 bits in UBRRH*/
300       UBRRH = BAUD_PRESCALE>>8;
301       UBRRL = BAUD_PRESCALE;
302  }
303
304  void UART_sendByte(const uint8 data)
305  {
306       /* UDRE flag is set when the Tx buffer (UDR) is empty and ready for
307        * transmitting a new byte so wait until this flag is set to one */
308       while(BIT_IS_CLEAR(UCSRA,UDRE)){}
309       /* Put the required data in the UDR register and it also clear the UDRE flag as
310        * the UDR register is not empty now */
311       UDR = data;
312       /************************* Another Method *************************
313       UDR = data;
314       while(BIT_IS_CLEAR(UCSRA,TXC)){} // Wait until the transimission is complete TXC = 1
315       SET_BIT(UCSRA,TXC); // Clear the TXC flag
316       ************************************************************************/
317  }
```

```c

uint8 UART_recieveByte(void)
{
	/* RXC flag is set when the UART receive data so wait until this
	 * flag is set to one */
	while(BIT_IS_CLEAR(UCSRA,RXC)){}
	/* Read the received data from the Rx buffer (UDR) and the RXC flag
	   will be cleared after read this data */
	return UDR;
}

void UART_sendString(const uint8 *Str)
{
	uint8 i = 0;
	while(Str[i] != '\0')
	{
		UART_sendByte(Str[i]);
		i++;
	}
	/************************* Another Method *************************
	while(*Str != '\0')
	{
		UART_sendByte(*Str);
		Str++;
	}
	*************************************************************/
}

void UART_receiveString(uint8 *Str)
{
	uint8 i = 0;
	Str[i] = UART_recieveByte();
	while(Str[i] != '#')
	{
		i++;
		Str[i] = UART_recieveByte();
	}
	Str[i] = '\0';
```

```
}
/*****************************************************************************
 *
 * Module: Micro - Configuration
 *
 * File Name: Micro_Config.h
 *
 * Description: File include all Microcontroller libraries
 *
 *****************************************************************************/

#ifndef MICRO_CONFIG_H_
#define MICRO_CONFIG_H_

#ifndef F_CPU
#define F_CPU 8000000UL //1MHz Clock frequency
#endif

#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

#endif /* MICRO_CONFIG_H_ */
/*****************************************************************************
 *
 * Module: Common - Macros
 *
 * File Name: Common_Macros.h
 *
 * Description: Commonly used Macros
 *
 * Author: Mohamed Tarek
 *

*****************************************************************************/

```

```c
#ifndef COMMON_MACROS
#define COMMON_MACROS

/* Set a certain bit in any register */
#define SET_BIT(REG,BIT) (REG|=(1<<BIT))

/* Clear a certain bit in any register */
#define CLEAR_BIT(REG,BIT) (REG&=(~(1<<BIT)))

/* Toggle a certain bit in any register */
#define TOGGLE_BIT(REG,BIT) (REG^=(1<<BIT))

/* Rotate right the register value with specific number of rotates */
#define ROR(REG,num) ( REG= (REG>>num) | (REG<<(8-num)) )

/* Rotate left the register value with specific number of rotates */
#define ROL(REG,num) ( REG= (REG<<num) | (REG>>(8-num)) )

/* Check if a specific bit is set in any register and return true if yes */
#define BIT_IS_SET(REG,BIT) ( REG & (1<<BIT) )

/* Check if a specific bit is cleared in any register and return true if yes */
#define BIT_IS_CLEAR(REG,BIT) ( !(REG & (1<<BIT)) )

#endif
/*****************************************************************************
 *
 * Module: Common - Platform Types Abstraction
 *
 * File Name: std_types.h
 *
 * Description: types for AVR
 *
```

```
**********************************************************
**********************/

#ifndef STD_TYPES_H_
#define STD_TYPES_H_

/* Boolean Data Type */
typedef unsigned char bool;

/* Boolean Values */
#ifndef FALSE
#define FALSE       (0u)
#endif
#ifndef TRUE
#define TRUE        (1u)
#endif

#define HIGH        (1u)
#define LOW         (0u)

typedef unsigned char         uint8;          /*
  0 .. 255             */
typedef signed char           sint8;          /*
  -128 .. +127            */
typedef unsigned short        uint16;         /*
  0 .. 65535           */
typedef signed short          sint16;         /*
  -32768 .. +32767          */
typedef unsigned long         uint32;         /*
  0 .. 4294967295      */
typedef signed long           sint32;         /* -
  2147483648 .. +2147483647     */
typedef unsigned long long    uint64;         /*
  0..18446744073709551615  */
typedef signed long long      sint64;
typedef float                 float32;
typedef double                float64;

#endif /* STD_TYPE_H_ */
/*
 * mfrc522_cmd.h
```

```
457   *
458   * Copyright 2013 Shimon <shimon@monistit.com>
459   *
460   * This program is free software; you can
   redistribute it and/or modify
461   * it under the terms of the GNU General Public
   License as published by
462   * the Free Software Foundation; either version 2 of
   the License, or
463   * (at your option) any later version.
464   *
465   * This program is distributed in the hope that it
   will be useful,
466   * but WITHOUT ANY WARRANTY; without even the implied
   warranty of
467   * MERCHANTABILITY or FITNESS FOR A PARTICULAR
   PURPOSE.  See the
468   * GNU General Public License for more details.
469   *
470   * You should have received a copy of the GNU General
   Public License
471   * along with this program; if not, write to the Free
   Software
472   * Foundation, Inc., 51 Franklin Street, Fifth Floor,
   Boston,
473   * MA 02110-1301, USA.
474   *
475   *
476   */
477
478  #ifndef MFRC522_CMD_H_
479  #define MFRC522_CMD_H_
480
481  //command set
482  #define Idle_CMD                 0x00
483  #define Mem_CMD                      0x01
484  #define GenerateRandomId_CMD0x02
485  #define CalcCRC_CMD                  0x03
486  #define Transmit_CMD             0x04
487  #define NoCmdChange_CMD          0x07
488  #define Receive_CMD                  0x08
489  #define Transceive_CMD           0x0C
```

```
#define Reserved_CMD            0x0D
#define MFAuthent_CMD           0x0E
#define SoftReset_CMD           0x0F


#endif /* MFRC522_CMD_H_ */
/*
 * mfrc522_reg.h
 *
 * Copyright 2013 Shimon <shimon@monistit.com>
 *
 * This program is free software; you can redistribute it
and/or modify
 * it under the terms of the GNU General Public License as
published by
 * the Free Software Foundation; either version 2 of the
License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be
useful,
 * but WITHOUT ANY WARRANTY; without even the implied
warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General
Public License
 * along with this program; if not, write to the Free
Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor,
Boston,
 * MA 02110-1301, USA.
 *
 *
 */


#ifndef MFRC522_REG_H_
#define MFRC522_REG_H_
```

```
//Page 0 ==> Command and Status
#define Page0_Reserved_1    0x00
#define CommandReg          0x01
#define ComIEnReg           0x02
#define DivIEnReg           0x03
#define ComIrqReg           0x04
#define DivIrqReg           0x05
#define ErrorReg            0x06
#define Status1Reg          0x07
#define Status2Reg          0x08
#define FIFODataReg              0x09
#define FIFOLevelReg        0x0A
#define WaterLevelReg       0x0B
#define ControlReg          0x0C
#define BitFramingReg       0x0D
#define CollReg                  0x0E
#define Page0_Reserved_2    0x0F

//Page 1 ==> Command
#define Page1_Reserved_1    0x10
#define ModeReg                  0x11
#define TxModeReg           0x12
#define RxModeReg           0x13
#define TxControlReg        0x14
#define TxASKReg            0x15
#define TxSelReg            0x16
#define RxSelReg            0x17
#define RxThresholdReg      0x18
#define  DemodReg           0x19
#define Page1_Reserved_2    0x1A
#define Page1_Reserved_3    0x1B
#define MfTxReg                  0x1C
#define MfRxReg                  0x1D
#define Page1_Reserved_4    0x1E
#define SerialSpeedReg      0x1F

//Page 2 ==> CFG
#define Page2_Reserved_1    0x20
#define CRCResultReg_1      0x21
#define CRCResultReg_2      0x22
#define Page2_Reserved_2    0x23
```

```
#define ModWidthReg             0x24
#define Page2_Reserved_3    0x25
#define RFCfgReg            0x26
#define GsNReg              0x27
#define CWGsPReg            0x28
#define ModGsPReg           0x29
#define TModeReg            0x2A
#define TPrescalerReg       0x2B
#define TReloadReg_1        0x2C
#define TReloadReg_2        0x2D
#define TCounterValReg_1    0x2E
#define TCounterValReg_2    0x2F

//Page 3 ==> TestRegister
#define Page3_Reserved_1    0x30
#define TestSel1Reg             0x31
#define TestSel2Reg             0x32
#define TestPinEnReg        0x33
#define TestPinValueReg     0x34
#define TestBusReg          0x35
#define AutoTestReg             0x36
#define VersionReg          0x37
#define AnalogTestReg       0x38
#define TestDAC1Reg             0x39
#define TestDAC2Reg             0x3A
#define TestADCReg          0x3B
#define Page3_Reserved_2    0x3C
#define Page3_Reserved_3    0x3D
#define Page3_Reserved_4    0x3E
#define Page3_Reserved_5    0x3F


#endif /* MFRC522_REG_H_ */
/*
 * mfrc522.h
 *
 * Copyright 2013 Shimon <shimon@monistit.com>
 *
 * This program is free software; you can redistribute it
and/or modify
```

```
 * it under the terms of the GNU General Public License as
published by
 * the Free Software Foundation; either version 2 of the
License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be
useful,
 * but WITHOUT ANY WARRANTY; without even the implied
warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General
Public License
 * along with this program; if not, write to the Free
Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor,
Boston,
 * MA 02110-1301, USA.
 *
 *
 */

#ifndef MFRC522_H_
#define MFRC522_H_

#include <stdint.h>

#include "mfrc522_cmd.h"
#include "mfrc522_reg.h"

#define CARD_FOUND      1
#define CARD_NOT_FOUND  2
#define ERROR           3

#define MAX_LEN             16

//Card types
#define Mifare_UltraLight   0x4400
#define Mifare_One_S50      0x0400
```

```
#define Mifare_One_S70      0x0200
#define Mifare_Pro_X        0x0800
#define Mifare_DESFire      0x4403

// Mifare_One card command word
# define PICC_REQIDL          0x26                // find
the antenna area does not enter hibernation
# define PICC_REQALL          0x52                // find
all the cards antenna area
# define PICC_ANTICOLL        0x93                // anti-
collision
# define PICC_SElECTTAG       0x93                //
election card
# define PICC_AUTHENT1A       0x60                //
authentication key A
# define PICC_AUTHENT1B       0x61                //
authentication key B
# define PICC_READ            0x30                // Read
Block
# define PICC_WRITE           0xA0                // write
block
# define PICC_DECREMENT       0xC0                // debit
# define PICC_INCREMENT       0xC1                //
recharge
# define PICC_RESTORE         0xC2                //
transfer block data to the buffer
# define PICC_TRANSFER        0xB0                // save
the data in the buffer
# define PICC_HALT            0x50                // Sleep

void mfrc522_init();
void mfrc522_reset();
void mfrc522_write(uint8_t reg, uint8_t data);
uint8_t mfrc522_read(uint8_t reg);
uint8_t  mfrc522_request(uint8_t req_mode, uint8_t *
tag_type);
uint8_t mfrc522_to_card(uint8_t cmd, uint8_t *send_data,
uint8_t send_data_len, uint8_t *back_data, uint32_t
*back_data_len);
```

```
#endif /* MFRC522_H_ */
/*
 * mfrc522.c
 *
 * Copyright 2013 Shimon <shimon@monistit.com>
 *
 * This program is free software; you can redistribute it
and/or modify
 * it under the terms of the GNU General Public License as
published by
 * the Free Software Foundation; either version 2 of the
License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be
useful,
 * but WITHOUT ANY WARRANTY; without even the implied
warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General
Public License
 * along with this program; if not, write to the Free
Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor,
Boston,
 * MA 02110-1301, USA.
 *
 *
 */
#include "mfrc522.h"
#include "spi.h"



void mfrc522_init()
{
    uint8_t byte;
    mfrc522_reset();
```

```
     mfrc522_write(TModeReg, 0x8D);
    mfrc522_write(TPrescalerReg, 0x3E);
    mfrc522_write(TReloadReg_1, 30);
    mfrc522_write(TReloadReg_2, 0);
     mfrc522_write(TxASKReg, 0x40);
     mfrc522_write(ModeReg, 0x3D);

     byte = mfrc522_read(TxControlReg);
     if(!(byte&0x03))
     {
          mfrc522_write(TxControlReg,byte|0x03);
     }
}

void mfrc522_write(uint8_t reg, uint8_t data)
{
     ENABLE_CHIP();
     spi_transmit((reg<<1)&0x7E);
     spi_transmit(data);
     DISABLE_CHIP();
}

uint8_t mfrc522_read(uint8_t reg)
{
     uint8_t data;
     ENABLE_CHIP();
     spi_transmit(((reg<<1)&0x7E)|0x80);
     data = spi_transmit(0x00);
     DISABLE_CHIP();
     return data;
}

void mfrc522_reset()
{
     mfrc522_write(CommandReg,SoftReset_CMD);
}

uint8_t  mfrc522_request(uint8_t req_mode, uint8_t *
tag_type)
{
     uint8_t  status;
```

```
    uint32_t backBits;//The received data bits

    mfrc522_write(BitFramingReg, 0x07);//TxLastBists =
BitFramingReg[2..0]     ???

    tag_type[0] = req_mode;
    status = mfrc522_to_card(Transceive_CMD, tag_type, 1,
tag_type, &backBits);

    if ((status != CARD_FOUND) || (backBits != 0x10))
    {
        status = ERROR;
    }

    return status;
}

uint8_t mfrc522_to_card(uint8_t cmd, uint8_t *send_data,
uint8_t send_data_len, uint8_t *back_data, uint32_t
*back_data_len)
{
    uint8_t status = ERROR;
    uint8_t irqEn = 0x00;
    uint8_t waitIRq = 0x00;
    uint8_t lastBits;
    uint8_t n;
    uint8_t   tmp;
    uint32_t i;

    switch (cmd)
    {
        case MFAuthent_CMD:      //Certification cards
close
        {
            irqEn = 0x12;
            waitIRq = 0x10;
            break;
        }
        case Transceive_CMD:    //Transmit FIFO data
        {
            irqEn = 0x77;
            waitIRq = 0x30;
```

```
                break;
            }
            default:
                break;
        }

        //mfrc522_write(ComIEnReg, irqEn|0x80);    //Interrupt
request
        n=mfrc522_read(ComIrqReg);
        mfrc522_write(ComIrqReg,n&(~0x80));//clear all
interrupt bits
        n=mfrc522_read(FIFOLevelReg);
        mfrc522_write(FIFOLevelReg,n|0x80);//flush FIFO data

         mfrc522_write(CommandReg, Idle_CMD);  //NO action;
Cancel the current cmd???

         //Writing data to the FIFO
        for (i=0; i<send_data_len; i++)
        {
            mfrc522_write(FIFODataReg, send_data[i]);
         }

         //Execute the cmd
         mfrc522_write(CommandReg, cmd);
        if (cmd == Transceive_CMD)
        {
            n=mfrc522_read(BitFramingReg);
            mfrc522_write(BitFramingReg,n|0x80);
         }

         //Waiting to receive data to complete
         i = 2000; //i according to the clock frequency
adjustment, the operator M1 card maximum waiting time
25ms???
        do
        {
            //CommIrqReg[7..0]
            //Set1 TxIRq RxIRq IdleIRq HiAlerIRq LoAlertIRq
ErrIRq TimerIRq
            n = mfrc522_read(ComIrqReg);
            i--;
```

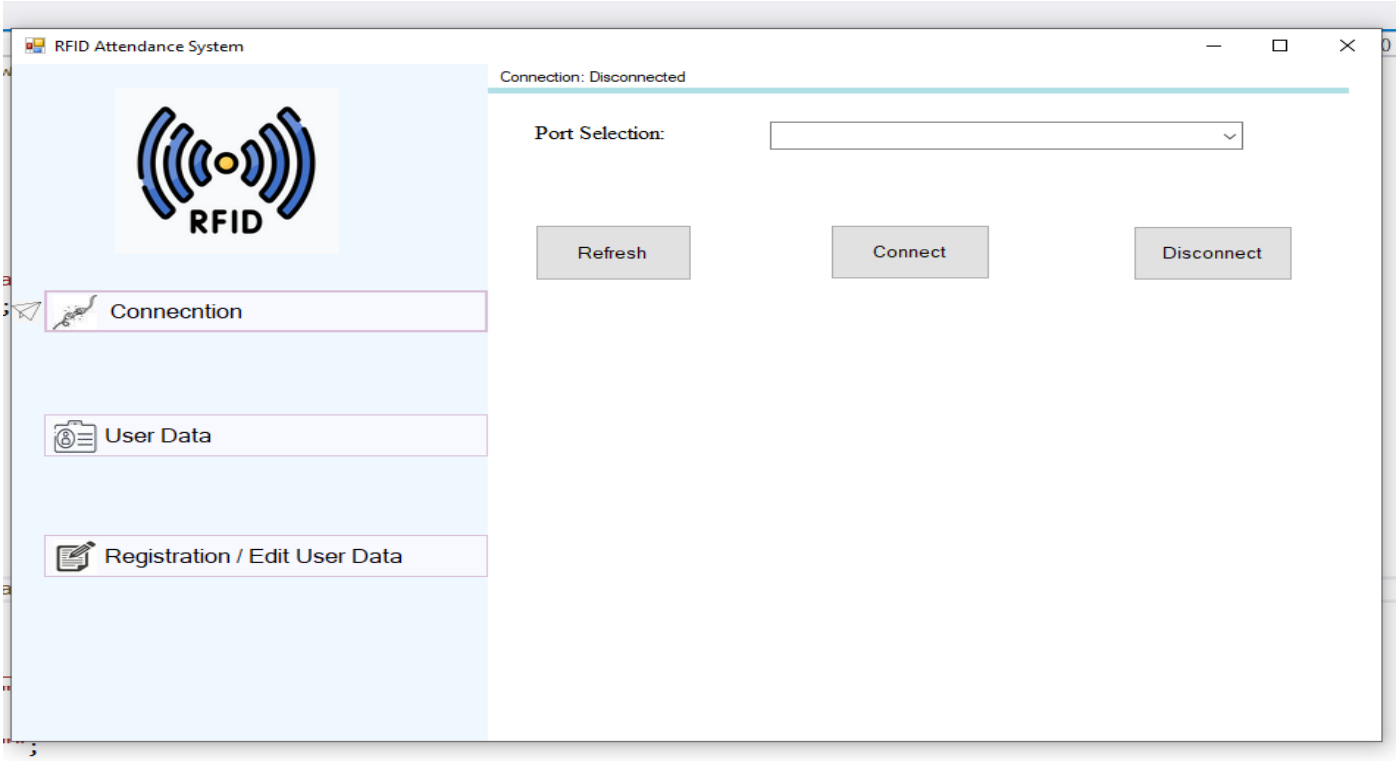
```
    }
    while ((i!=0) && !(n&0x01) && !(n&waitIRq));

     tmp=mfrc522_read(BitFramingReg);
     mfrc522_write(BitFramingReg,tmp&(~0x80));

    if (i != 0)
    {
        if(!(mfrc522_read(ErrorReg) & 0x1B))
     //BufferOvfl Collerr CRCErr ProtecolErr
        {
            status = CARD_FOUND;
            if (n & irqEn & 0x01)
            {
                   status = CARD_NOT_FOUND;
     //??
              }

            if (cmd == Transceive_CMD)
            {
                   n = mfrc522_read(FIFOLevelReg);
              lastBits = mfrc522_read(ControlReg) & 0x07;
                if (lastBits)
                {
                       *back_data_len = (n-1)*8 +
lastBits;
                   }
                else
                {
                       *back_data_len = n*8;
                   }

                if (n == 0)
                {
                       n = 1;
                   }
                if (n > MAX_LEN)
                {
                       n = MAX_LEN;
                   }

                   //Reading the received data in FIFO
```

```
                for (i=0; i<n; i++)
                {
                        back_data[i] =
mfrc522_read(FIFODataReg);
                }
            }
        }
        else
        {
              status = ERROR;
         }

    }

    //SetBitMask(ControlReg,0x80);           //timer stops
    //mfrc522_write(cmdReg, PCD_IDLE);

    return status;
}
```

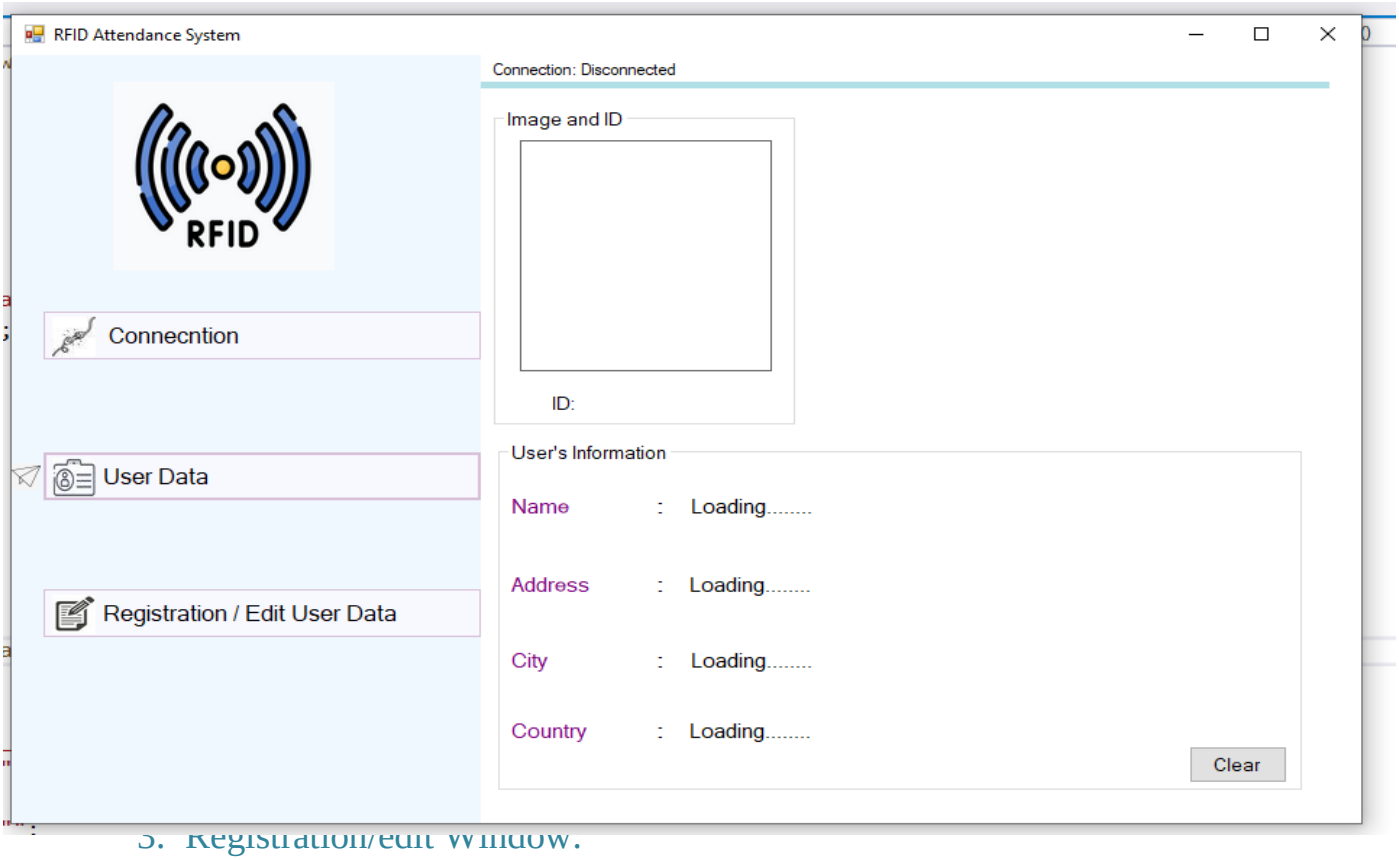
2. PC-side source code:

1) GUI Application:

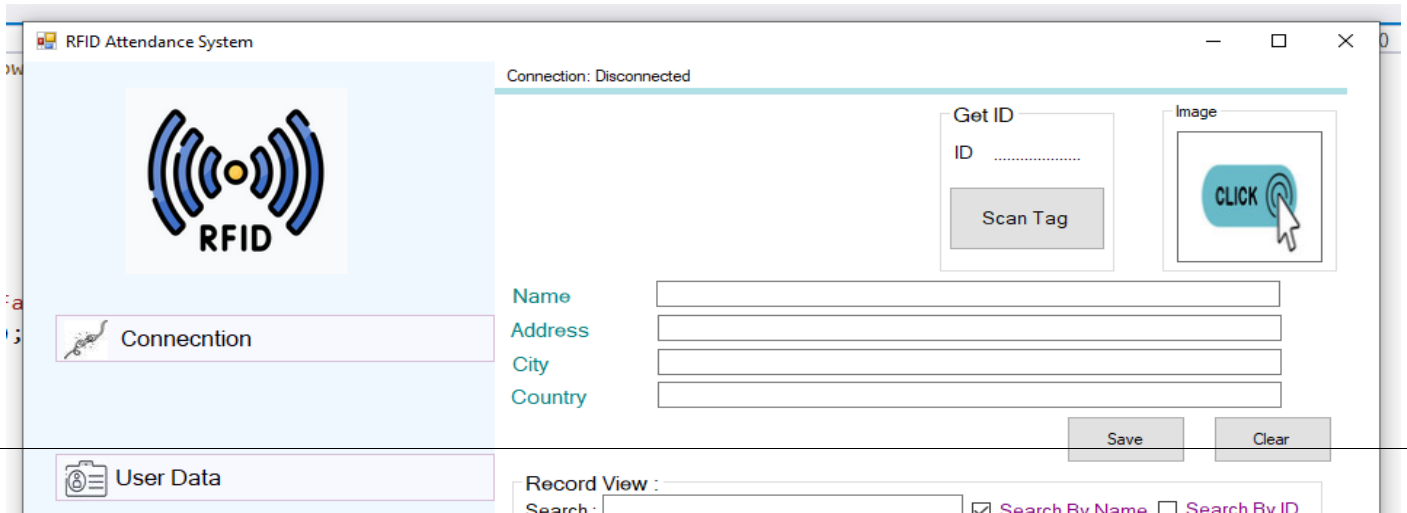
1. Connection Window:



2. User Data Window:



3. Registration/edit Window:



2) GUI Source Code:

```
using System;
using System.IO;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO.Ports;
using MySql.Data.MySqlClient;


namespace RFID_Attendence_System_B1
{
   public partial class Form1 : Form
   {

       private MySqlConnection Connection = new MySqlConnection("server=localhost;
user=root; password=; database=rfid_user_data");
       private MySqlCommand MySQLCMD = new MySqlCommand();
       private MySqlDataAdapter MySQLDA = new MySqlDataAdapter();
       private DataTable DT = new DataTable();
       private string Table_Name = "rfid_user_data_table"; // your table name
       private int Data;
       int idrecieved = 0;
       private bool LoadImagesStr = false;
       private string IDRam;
       private string IMG_FileNameInput;
       private string StatusInput = "Save";
       private string SqlCmdSearchstr;
```

```
        public static string StrSerialIn;
        private bool GetID = false;
        private bool ViewUserData = false;

        SerialPort port = null;
        string data_rx = "";
        bool flag_st = false;
        public Form1()
        {
            InitializeComponent();
            refresh_com();
        }
        private void Form1_Load(object sender, EventArgs e)
        {

            PanelRegisterationandEditUserData.Visible = false;
            paneluserdata.Visible = false;
            panelconn.Visible = true;

        }
        private void panel1_Paint(object sender, PaintEventArgs e)
        {

        }

        private void label1_Click(object sender, EventArgs e)
        {

        }

        private void connectionbutton_Click(object sender, EventArgs e)
        {
            pictureBox2.Top = connectionbutton.Top;
            PanelRegisterationandEditUserData.Visible = false;
            paneluserdata.Visible = false;
            panelconn.Visible = true;
        }

        private void userdatabutton_Click(object sender, EventArgs e)
        {
            if (TimerSerialIn.Enabled == false)
            {
                MessageBox.Show("Failed to open User Data !!!Click the Connection menu then click
the Connect button.", "Information");
                MessageBox.Show("Connection failed !!!Please check that the server is ready !!!",
"Error Message", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
```

```
    else
    {
        StrSerialIn = "";
        ViewUserData = true;
        pictureBox2.Top = userdatabutton.Top;
        PanelRegisterationandEditUserData.Visible = false;
        paneluserdata.Visible = true;
        panelconn.Visible = false;
        ShowDataUser();
   }
}

private void regbutton_Click(object sender, EventArgs e)
{
    StrSerialIn = "";
    ViewUserData = false;



    pictureBox2.Top = regbutton.Top;
    PanelRegisterationandEditUserData.Visible = true;
    paneluserdata.Visible = false;
    panelconn.Visible = false;



    ShowData();
}
private void connect()
{
    port = new SerialPort(comboBox1.SelectedItem.ToString());
  //  port.DataReceived += new SerialDataReceivedEventHandler(TimerSerialIn_Tick);

    port.BaudRate = 9600;
    port.DataBits = 8;
    port.StopBits = StopBits.One;

    try
    {
        if (!port.IsOpen)
        {
            port.Open();
            MessageBox.Show("Open");
            TimerSerialIn.Start();
        }
    }
    catch (Exception ex)
```

```
    {

    }
}
private void disconnect()
{
    try
    {
        if (port.IsOpen)
        {
            port.Close();
            MessageBox.Show("Close");
            label1connstatus.Text = "Connection Stauts : Disconnected";
            label1connstatus.ForeColor = Color.Red;
            TimerSerialIn.Stop();
        }
    }
    catch (Exception ex)
    {

    }
}
private void refresh_com()
{
    comboBox1.DataSource = SerialPort.GetPortNames();
}
private void DataReceiveHandler(object sender, SerialDataReceivedEventArgs e)
{
    SerialPort sp = (SerialPort)sender;
    string indata = "";
    try
    {
        indata = sp.ReadExisting();
    }
    catch (Exception E) { }
    int idx_end = indata.IndexOf(';');
    if ((idx_end >= 0) && (flag_st == true))
    {
        flag_st = false;
        data_rx += indata.Substring(0, idx_end);
    }

    int idx_start = indata.IndexOf('@');
    if (idx_start >= 0)
    {
        flag_st = true;
        data_rx = "";
        if (indata.Length > (idx_start + 1))
```

```
            {
                data_rx += indata.Substring((idx_start + 1), (indata.Length - 1));
                int idx = data_rx.IndexOf(';');
                if (idx >= 0)
                {
                    data_rx = data_rx.Substring(0, idx);
                }
            }
        }
        if (flag_st)
        {
            data_rx += indata;
        }





    }
    private void ShowData()
    {
        try
        { Connection.Open();
        }
        catch (Exception ex)
        { MessageBox.Show("Connection failed !!!Please888check that the server is ready !!!",
"Error Message", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }

        try
        {
            if (LoadImagesStr == false)
            {
                MySQLCMD.CommandType = CommandType.Text;
                MySQLCMD.CommandText = "SELECT * FROM " + Table_Name + " WHERE ID
LIKE '" + labelid.Text.Substring(5, labelid.Text.Length - 5) + "'";
                MySQLDA = new MySqlDataAdapter(MySQLCMD.CommandText, Connection);
                DT = new DataTable();
                Data = MySQLDA.Fill(DT);

                if (Data > 0)
                {
                    dataGridView1.DataSource = null;
                    dataGridView1.DataSource = DT;
                    dataGridView1.SelectedColumns[2].DefaultCellStyle.Format = "c";
                    dataGridView1.DefaultCellStyle.ForeColor = Color.Black;
                    dataGridView1.ClearSelection();
```

```
                }
                else
                { dataGridView1.DataSource = DT; }

            }
            else
            {
                MySQLCMD.CommandType = CommandType.Text;
                MySQLCMD.CommandText = "SELECT Images FROM " + Table_Name + "
WHERE ID LIKE '" + IDRam + "'";
                MySQLDA = new MySqlDataAdapter(MySQLCMD.CommandText, Connection);
                DT = new DataTable();
                Data = MySQLDA.Fill(DT);


                DataRow row = DT.Rows[0];

                if (Data > 0)
                {

                    byte[] ImgArray = (byte[])row["Images"];

                    var lmgStr = new MemoryStream(ImgArray);
                    PictureBoxImagePreview.Image = Image.FromStream(lmgStr);
                    PictureBoxImagePreview.SizeMode = PictureBoxSizeMode.Zoom;
                    lmgStr.Close();
                  //}
                }

                LoadImagesStr = false;
            }

        }
        catch (Exception ex)
        {
            MessageBox.Show("Failed to load Database !!!" + ex.Message, "Error Message");
            Connection.Close();
            return;
        }



        DT = null;
        Connection.Close();
    }

    private void ShowDataUser()
    {
```

```
            try
            {
                Connection.Open();
            }

            catch (Exception ex)
            {
                MessageBox.Show("Show data userConnection failed !!! Please check that the server is
ready !!!", "Error Message", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            try
            {
               // idrecieved = Int32.Parse(StrSerialIn);
                MySQLCMD.CommandType = CommandType.Text;
                MySQLCMD.CommandText = "SELECT * FROM " + Table_Name + " WHERE ID
LIKE '"  + labelid.Text.Substring(5,labelid.Text.Length-5) + "'";
                MySQLDA = new MySqlDataAdapter(MySQLCMD.CommandText, Connection);
                DT = new DataTable();
                Data = MySQLDA.Fill(DT);
                if (Data > 0)
                {
                    byte[] ImgArray = (byte[])DT.Rows[0]["Images"];
                    MemoryStream lmgStr = new MemoryStream(ImgArray);

                    PictureBoxImagePreview.Image = Image.FromStream(lmgStr);

                    lmgStr.Close();

                    labelid.Text = DT.Rows[0]["ID"].ToString();
                    LabelName.Text = DT.Rows[0]["Name"].ToString();
                    LabelAddress.Text = DT.Rows[0]["Address"].ToString();
                    LabelCity.Text = DT.Rows[0]["City"].ToString();
                    LabelCountry.Text = DT.Rows[0]["Country"].ToString();
                }

                else
                {
                    MessageBox.Show("ID not found !!! Please register your ID.", "Information
Message");
                }


            }

            catch (Exception ex)
            {
```

```
            MessageBox.Show("Show data user Failed to load Database !!!" + ex.Message, "Error
Message");
            Connection.Close();
            return;
        }



        DT = null;
        Connection.Close();
    }

    private void ClearInputUpdateData()
    {
        TextBoxName.Text = "";
        LabelGetID.Text = "________";
        TextBoxAddress.Text = "";
        TextBoxCity.Text = "";
        TextBoxCountry.Text = "";
        PictureBoxImageInput.Image =
Properties.Resources.rsz_1click_here_button_with_cursor_icon_vector_23394642;
    }


    /*private void data_rx_handler(object sender, SerialDataReceivedEventArgs e)
    {
        SerialPort sp = (SerialPort)sender;
        string temp = sp.ReadExisting();
        if (temp.Contains("@"))
        {
            data_rx = "";
            flag_st = true;
        }
        else if (temp.Contains(";"))
        {
            flag_st = false;
            MessageBox.Show(data_rx);

        }
        else if (temp.Contains("%"))
        {
            //whatever.
        }
        if (flag_st)
        {
            data_rx += temp;
        }
    }
```

```
*/

private void refreshbutton_Click(object sender, EventArgs e)
{
    refresh_com();
}

private void connectbutton_Click(object sender, EventArgs e)
{
    connect();
}

private void disconnectb_Click(object sender, EventArgs e)
{
    disconnect();
}

private void pictureBox1_Click(object sender, EventArgs e)
{

}

private void PanelRegisterationandEditUserData_Paint(object sender, PaintEventArgs e)
{

}

private void groupBox2_Enter(object sender, EventArgs e)
{

}

private void groupBox3_Enter(object sender, EventArgs e)
{

}

private void panelconn_Paint(object sender, PaintEventArgs e)
{

}

private void ButtonSave_Click(object sender, EventArgs e)
{


    byte[] arrImage;
    MemoryStream mstream = new MemoryStream();
```

```
        if (TextBoxName.Text == "")

        {
            MessageBox.Show("Name cannot be empty !!!", "Error Message",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }

        if (TextBoxAddress.Text == "")
        {
            MessageBox.Show("Address cannot be empty !!!", "Error Message",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }

        if (TextBoxCity.Text == "")
        {
            MessageBox.Show("City cannot be empty !!!", "Error Message",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }

        if (TextBoxCountry.Text == "")
        {
            MessageBox.Show("Country cannot be empty !!!", "Error Message",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }



        if (StatusInput == "Save")
        {
            if (IMG_FileNameInput != "")
            {
                PictureBoxImageInput.Image.Save(mstream,
System.Drawing.Imaging.ImageFormat.Jpeg);
                arrImage = mstream.GetBuffer();
            }
            else
            {
                MessageBox.Show("The image has not been selected !!!", "Error Message",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            try
```

```
            {
                Connection.Open();
            }

            catch (Exception ex)
            {
                MessageBox.Show("Connection failed !!! Please check that the server is ready !!!",
"Error Message", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            try
            {
                MySQLCMD = new MySqlCommand();
                {
                    var withBlock = MySQLCMD;
                    withBlock.CommandText = "INSERT INTO " + Table_Name + " (Name, ID,
Address, City, Country, Images) VALUES (@name, @ID, @address, @city, @country,
@images)";
                    withBlock.Connection = Connection;
                    withBlock.Parameters.AddWithValue("@name", TextBoxName.Text);
                    withBlock.Parameters.AddWithValue("@id", LabelGetID.Text);
                    withBlock.Parameters.AddWithValue("@address", TextBoxAddress.Text);
                    withBlock.Parameters.AddWithValue("@city", TextBoxCity.Text);
                    withBlock.Parameters.AddWithValue("@country", TextBoxCountry.Text);
                    withBlock.Parameters.AddWithValue("@images", arrImage);
                    withBlock.ExecuteNonQuery();
                }

                MessageBox.Show("Data saved successfully", "Information");
                IMG_FileNameInput = "";
                ClearInputUpdateData();
            }

            catch (Exception ex)
            {
                MessageBox.Show("Data failed to save !!!" + ex.Message, "Error Message");
                Connection.Close();
                return;
            }


            Connection.Close();
        }

        else
        {
            if (IMG_FileNameInput != "")
```

```
            {
                PictureBoxImageInput.Image.Save(mstream,
System.Drawing.Imaging.ImageFormat.Jpeg);
                arrImage = mstream.GetBuffer();



                try
                {
                    Connection.Open();
                }

                catch (Exception ex)
                {
                    MessageBox.Show("Connection failed !!! Please check that the server is ready
!!!", "Error Message", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    return;
                }


                try
                {
                    MySQLCMD = new MySqlCommand();
                    {
                        var withBlock = MySQLCMD;
                        withBlock.CommandText = "UPDATE " + Table_Name + " SET
Name=@name,ID=@id,Address=@address,City=@city,Country=@country,Images=@images
WHERE ID=@id ";
                        withBlock.Connection = Connection;
                        withBlock.Parameters.AddWithValue("@name", TextBoxName.Text);
                        withBlock.Parameters.AddWithValue("@id", LabelGetID.Text);
                        withBlock.Parameters.AddWithValue("@address", TextBoxAddress.Text);
                        withBlock.Parameters.AddWithValue("@city", TextBoxCity.Text);
                        withBlock.Parameters.AddWithValue("@country", TextBoxCountry.Text);
                        withBlock.Parameters.AddWithValue("@images", arrImage);
                        withBlock.ExecuteNonQuery();
                    }

                    MessageBox.Show("Data updated successfully", "Information");
                    IMG_FileNameInput = "";
                    ButtonSave.Text = "Save";
                    ClearInputUpdateData();
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Data failed to Update !!!" + ex.Message, "Error Message");
                    Connection.Close();
                    return;
```

```
            }


            Connection.Close();
        }
        else
        {
            try
            {
                Connection.Open();
            }

            catch (Exception ex)
            {
                MessageBox.Show("Connection failed !!! Please check that the server is ready
!!!", "Error Message", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }


            try
            {
                MySQLCMD = new MySqlCommand();
                {
                    var withBlock = MySQLCMD;
                    withBlock.CommandText = "UPDATE " + Table_Name + " SET
Name=@name,ID=@id,Address=@address,City=@city,Country=@country WHERE ID=@id ";
                    withBlock.Connection = Connection;
                    withBlock.Parameters.AddWithValue("@name", TextBoxName.Text);
                    withBlock.Parameters.AddWithValue("@id", LabelGetID.Text);
                    withBlock.Parameters.AddWithValue("@address", TextBoxAddress.Text);
                    withBlock.Parameters.AddWithValue("@city", TextBoxCity.Text);
                    withBlock.Parameters.AddWithValue("@country", TextBoxCountry.Text);
                    withBlock.ExecuteNonQuery();
                }

                MessageBox.Show("Data updated successfully", "Information");
                ButtonSave.Text = "Save";
                ClearInputUpdateData();
            }

            catch (Exception ex)
            {
                MessageBox.Show("Data failed to Update !!!" + ex.Message, "Error Message");
                Connection.Close();
                return;
            }
```

```
            Connection.Close();
        }



        StatusInput = "Save";
    }
    PictureBoxImagePreview.Image = null;
    ShowData();
}

private void buttonclear_Click(object sender, EventArgs e)
{
    labelid.Text = "ID : _________";
    LabelName.Text = "Waiting...";
    LabelAddress.Text = "Waiting...";
    LabelCity.Text = "Waiting...";
    LabelCountry.Text = "Waiting...";
    PictureBoxImagePreview.Image = null;
}

private void LabelName_Click(object sender, EventArgs e)
{

}

private void label1_Click_1(object sender, EventArgs e)
{

}

private void labelid_Click(object sender, EventArgs e)
{

}

private void ButtonClearForm_Click(object sender, EventArgs e)
{
    ClearInputUpdateData();
}

private void ButtonScanID_Click(object sender, EventArgs e)
{
    if (TimerSerialIn.Enabled == true)
    {
        PanelReadingTagProcess.Visible = true;
        GetID = true;
        ButtonScanID.Enabled = false;
```

```
            }

            else
            {
                MessageBox.Show("Failed to open User Data !!! Click the Connection menu then click
the Connect button.", "Error Message");
            }

        }

        private void PictureBoxImageInput_Click(object sender, EventArgs e)
        {
            OpenFileDialog OpenFileDialog1 = new OpenFileDialog();
            OpenFileDialog1.FileName = "";
            OpenFileDialog1.Filter = "JPEG (*.jpeg;*.jpg)|*.jpeg;*.jpg";

            if (OpenFileDialog1.ShowDialog(this) == System.Windows.Forms.DialogResult.OK)
            {
                IMG_FileNameInput = OpenFileDialog1.FileName;
                PictureBoxImageInput.ImageLocation = IMG_FileNameInput;
            }

        }

        private void CheckBoxByName_CheckedChanged(object sender, EventArgs e)
        {
            if (CheckBoxByName.Checked == true)
            {
                CheckBoxByID.Checked = false;
            }
            if (CheckBoxByName.Checked == false)
            {
                CheckBoxByID.Checked = true;
            }

        }

        private void CheckBoxByID_CheckedChanged(object sender, EventArgs e)
        {
            if (CheckBoxByID.Checked == true)
            {
                CheckBoxByName.Checked = false;
            }

            if (CheckBoxByID.Checked == false)
            {
                CheckBoxByName.Checked = true;
            }
```

```
        }

        private void label15_Click(object sender, EventArgs e)
        {

        }

        private void TextBoxSearch_TextChanged(object sender, EventArgs e)
        {
            if (CheckBoxByID.Checked == true)
            {
                if (TextBoxSearch.Text == null)
                {
                    SqlCmdSearchstr = "SELECT Name, ID, Address, City, Country FROM " +
Table_Name + " ORDER BY Name";
                }

                else
                {
                    SqlCmdSearchstr = "SELECT Name, ID, Address, City, Country FROM " +
Table_Name + " WHERE ID LIKE'" + TextBoxSearch.Text + "%'";
                }
            }
            if (CheckBoxByName.Checked == true)
            {
                if (TextBoxSearch.Text == null)
                {
                    SqlCmdSearchstr = "SELECT Name, ID, Address, City, Country FROM " +
Table_Name + " ORDER BY Name";
                }

                else
                {
                    SqlCmdSearchstr = "SELECT Name, ID, Address, City, Country FROM " +
Table_Name + " WHERE Name LIKE'" + TextBoxSearch.Text + "%'";
                }

            }
            try
            {
                Connection.Open();
            }

            catch (Exception ex)
            {
                MessageBox.Show("Connection failed !!! Please check that the server is ready !!!",
"Error Message", MessageBoxButtons.OK, MessageBoxIcon.Error);
```

```
            return;
        }

        try
        {
            MySQLDA = new MySqlDataAdapter(SqlCmdSearchstr, Connection);
            DT = new DataTable();
            Data = MySQLDA.Fill(DT);
            if (Data > 0)
            {
                dataGridView1.DataSource = null;
                dataGridView1.DataSource = DT;
                dataGridView1.DefaultCellStyle.ForeColor = Color.Black;
                dataGridView1.ClearSelection();
            }

            else
            {
                dataGridView1.DataSource = DT;
            }

        }

        catch (Exception ex)
        {
            MessageBox.Show("Failed to search" + ex.Message, "Error Message");
            Connection.Close();
        }
        Connection.Close();

    }

    private void dataGridView1_CellContentClick(object sender, DataGridViewCellEventArgs
e)
    {

    }

    private void dataGridView1_CellMouseDown(object sender,
DataGridViewCellMouseEventArgs e)
    {
        try
        {
            if (AllCellsSelected(dataGridView1) == false)
            {
                if (e.Button == MouseButtons.Left)
                {
```

```
                    // dataGridView1.CurrentCell =
this.songsDataGridView.Columns[e.ColumnIndex];
                    int i;
                    if (e.RowIndex >= 0)
                    {
                        i = dataGridView1.CurrentRow.Index;
                        LoadImagesStr = true;
                        IDRam = dataGridView1.Rows[i].Cells["ID"].Value.ToString();
                        ShowData();
                    }
                }

            }

        }

        catch (Exception ex)
        {
            return;
        }
    }
    private bool AllCellsSelected(DataGridView dgv)
    {
        bool AllCellsSelectedRet = default;
        AllCellsSelectedRet = dataGridView1.SelectedCells.Count == dataGridView1.RowCount
* dataGridView1.Columns.GetColumnCount(DataGridViewElementStates.Visible);
        return AllCellsSelectedRet;
    }


    /*private void TimerTimeDate_Tick(object sender, EventArgs e)
    {
        LabelDateTime.Text = "Time " & DateTime.Now.ToString("HH:mm:ss") & "  Date " &
DateTime.Now.ToString("dd MMM, yyyy");
    }*/

    private void deleteToolStripMenuItem_Click(object sender, EventArgs e)
    {
        if (dataGridView1.RowCount == 0)
        {
            MessageBox.Show("Cannot delete, table data is empty", "Error Message");/* TODO
ERROR: Skipped SkippedTokensTrivia */
            return;
        }
        if (dataGridView1.SelectedRows.Count == 0)
        {
            MessageBox.Show("Cannot delete, select the table data to be deleted", "Error
Message");
```

```
            return;
        }

        try
        {
            Connection.Open();
        }
        catch (Exception ex)
        {
            MessageBox.Show("Connection failed !!! Please check that the server is ready !!!",
"Error Message", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }

        try
        {
            if (AllCellsSelected(dataGridView1) == true)
            {
                MySQLCMD.CommandType = CommandType.Text;
                MySQLCMD.CommandText = "DELETE FROM " + Table_Name;
                MySQLCMD.Connection = Connection;
                MySQLCMD.ExecuteNonQuery();
            }
            foreach (DataGridViewRow row in dataGridView1.SelectedRows)
            {
                if (row.Selected == true)
                {

                    String Text = row.DataBoundItem.ToString();
                    MySQLCMD.CommandType = CommandType.Text;
                    MySQLCMD.CommandText = "DELETE FROM " + Table_Name + " WHERE
ID='" + Text + "'";
                    MySQLCMD.Connection = Connection;
                    MySQLCMD.ExecuteNonQuery();
                }
            }

        }



        catch (Exception ex)
        {
            MessageBox.Show("Failed to delete" + ex.Message, "Error Message");
            Connection.Close();
        }

        PictureBoxImagePreview.Image = null;
```

```
    Connection.Close();
    ShowData();
}

private void selectAllToolStripMenuItem_Click(object sender, EventArgs e)
{
    dataGridView1.SelectAll();
}

private void clearSelectionToolStripMenuItem_Click(object sender, EventArgs e)
{
    dataGridView1.ClearSelection();
    PictureBoxImagePreview.Image = null;
}

private void refreshToolStripMenuItem1_Click(object sender, EventArgs e)
{
    ShowData();
}

private void TimerSerialIn_Tick(object sender, EventArgs e)
{
    try
    {
        StrSerialIn = port.ReadExisting();
        //idrecieved = Int32.Parse(StrSerialIn);
        label1connstatus.Text = "Connection Status : Connected";
        label1connstatus.ForeColor = Color.Green;
        if (StrSerialIn != "")
        {
            if (GetID == true)
            {
                LabelGetID.Text = StrSerialIn;
                GetID = false;
                if (LabelGetID.Text != "________")
                {
                    PanelReadingTagProcess.Visible = false;
                    IDCheck();
                }
            }

            if (ViewUserData == true)
            {
                ViewData();
            }
        }
    }
    catch (Exception ex)
```

```
            {
                TimerSerialIn.Stop();
                disconnect();
                label1connstatus.Text = "Connection Status : Disconnect";
                //PictureBoxStatusConnect.Image = global::My.Resources.Disconnect;
                MessageBox.Show("Failed to connect !!!Microcontroller is not detected.", "Error
Message");
                connectbutton_Click(sender, e);
                return;
            }
        }
        private void IDCheck()
        {
            try
            {
                Connection.Open();
            }
            catch (Exception ex)
            {
                MessageBox.Show("ID CHeck Connection failed !!! Please check that the server is
ready !!!", "Error Message", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            try
            {
                MySQLCMD.CommandType = CommandType.Text;
                MySQLCMD.CommandText = "SELECT * FROM " + Table_Name + " WHERE ID
LIKE '" + LabelGetID.Text + "'";
                MySQLDA = new MySqlDataAdapter(MySQLCMD.CommandText, Connection);
                DT = new DataTable();
                Data = MySQLDA.Fill(DT);
                if (Data > 0)
                {
                    if (MessageBox.Show("ID registered ! Do you want to edit the data ?",
"Confirmation", MessageBoxButtons.OKCancel,MessageBoxIcon.Question) ==
DialogResult.Cancel)
                    {
                        DT = null;
                        Connection.Close();
                        ButtonScanID.Enabled = true;
                        GetID = false;
                        LabelGetID.Text = "________";
                        return;
                    }
                    else
                    {
                        byte[] ImgArray = (byte[])DT.Rows[0]["Images"];
```

```
                var lmgStr = new MemoryStream(ImgArray);
                PictureBoxImageInput.Image = Image.FromStream(lmgStr);
                PictureBoxImageInput.SizeMode = PictureBoxSizeMode.Zoom;
                TextBoxName.Text = DT.Rows[0]["ID"].ToString();

                TextBoxAddress.Text = DT.Rows[0]["Address"].ToString();
                TextBoxCity.Text = DT.Rows[0]["City"].ToString();
                TextBoxCountry.Text = DT.Rows[0]["Country"].ToString();
                StatusInput = "Update";
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Failed to load Database !!!" + ex.Message, "Error Message");
        Connection.Close();
        return;
    }

    DT = null;
    Connection.Close();
    ButtonScanID.Enabled = true;
    GetID = false;
}

private void ViewData()
{
    labelid.Text = "ID : " + StrSerialIn;
    if (labelid.Text == "ID : _________")
    {
        ViewData();
    }
    else
    {
        ShowDataUser();
    }
}

private void ButtonCloseReadingTag_Click(object sender, EventArgs e)
{
    PanelReadingTagProcess.Visible = false;
    ButtonScanID.Enabled = true;
}

private void groupBoxdetaileddata_Enter(object sender, EventArgs e)
{

}
```

```
    }
}
```