WORKSHOP project report FILE



VITH SEMESTER

Project report ON

RFID Based Attendance System

Submitted by:

Sidharth Mehta

# **Abstract**

Most educational institutions administrators are concerned about student irregular attendance. Truancies can affect student overall academic performance. The conventional method of taking attendance by calling names or signing on paper is very time consuming and insecure, hence inefficient. **Radio Frequency Identification (RFID)** is an interesting solution in such cases. RFID based attendance system is one of the solutions to address this problem.

# **Applications of RFID**

This system can be used to take attendance for student in school, college, and university. It also can be used to take attendance for workers in working places. Its ability to uniquely identify each person based on their RFID tag type of ID card make the process of taking the attendance easier, faster and secure as compared to conventional method.

The system can be connected to the computer through RS232 or Universal Serial Bus (USB) port and store the attendance taken inside database. Radio frequency technology is used in many applications. RFID tags are of two types

Passive tags contain 13-digit number tag inbuilt in it, whereas active tags are read/write tag i.e. one can read from the tag and write to the tag.

This project uses a passive tag in making the RFID based attendance system. In real time, one can issue active tags to the students, with their roll numbers as their tags. RFID reader contains a copper winding in it. This winding acts as an antenna.

# **Working of RFID based attendance system**

When a person with RFID tag or transponder enters in the range of RFID reader, the RF field induces voltage in the coils of tag. The range can be set by using the appropriate reader of appropriate frequency. This induced field supplies the voltage in case of passive tags and act as a battery in that case. If active tags are used then the case will be different as they have battery of their own.

Due to interaction of tag with reader 12 characters from tag are sent to controller. These 12 characters are sent to computer via serial communication from the controller. In computer data of employee are saved. In our case data of five employees are saved i.e. tag number and name.

When 12 characters are transferred to computer, a computer program matches the UID with the saved UIDs. If the UID is found then the program logs name of the employee corresponding to the UID, the time and date of entry in a text file. If the UID is not found, no action is taken. This way it can be done number of times for different employees.

# **Component List**

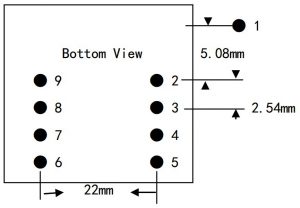
* Perforated PCB
* RFID Tags
* EM-18 RFID Reader
* Capacitor (100 µF and 0.01 µF)
* LED
* Connecting wires
* Resistors (2.2KΩ and 470Ω)
* Arduino Nano
* USB A to B cable
* Buzzer
* BC 557

# **RFID Reader**

The EM-18 RFID reader is a standalone module with RFID reader and antenna. It is very small (32mmx32mm) in size and easy to integrate with any hardware design. It supports 125 Kilohertz RFID tags and has Dual in line pins at 0.1-inch distance. Onboard antenna and hard plastic cover makes device small and sturdy. The module works on UART protocol which allows user to integrate it with any PC or Microcontroller based design. It also supports Wiegand protocol.



**Fig. 1 EM 18 RFID Reader**



**Fig. 2 EM-18 Pinout**

|  |  |  |
| --- | --- | --- |
| **Pin No.** | **Name** | **Function** |
| 1 | VCC | 5V |
| 2 | GND | Ground |
| 3 | BEEP | BEEP and LED |
| 4 | ANT | No Use |
| 5 | ANT | No Use |
| 6 | SEL | HIGH selects RS232, LOW selects WEIGAND |
| 7 | TX | UART TX, When RS232 is Selected |
| 8 | D1 | WIEGAND Data 1 |
| 9 | D0 | WIEGAND Data 0 |

**Table 1 Pinout description**

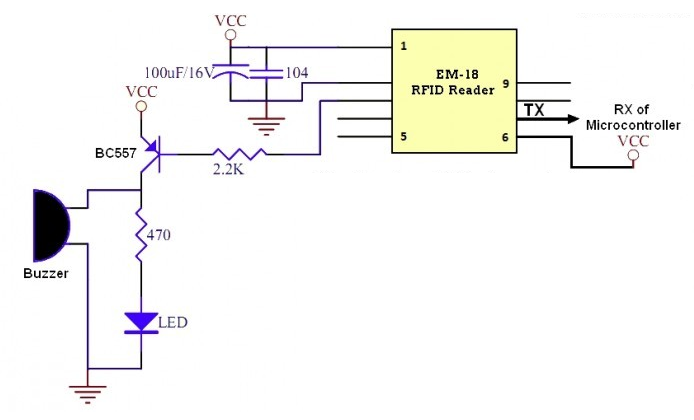
# **RFID Tag**

**These tags contain at least two parts**: an integrated circuit for storing and processing information, modulating and demodulating a radio-frequency (RF) signal, collecting DC power from the incident reader signal, and other specialized functions; and an antenna for receiving and transmitting the signal. The tag information is stored in a non-volatile memory. The RFID tag includes either fixed or programmable logic for processing the transmission and sensor data, respectively.

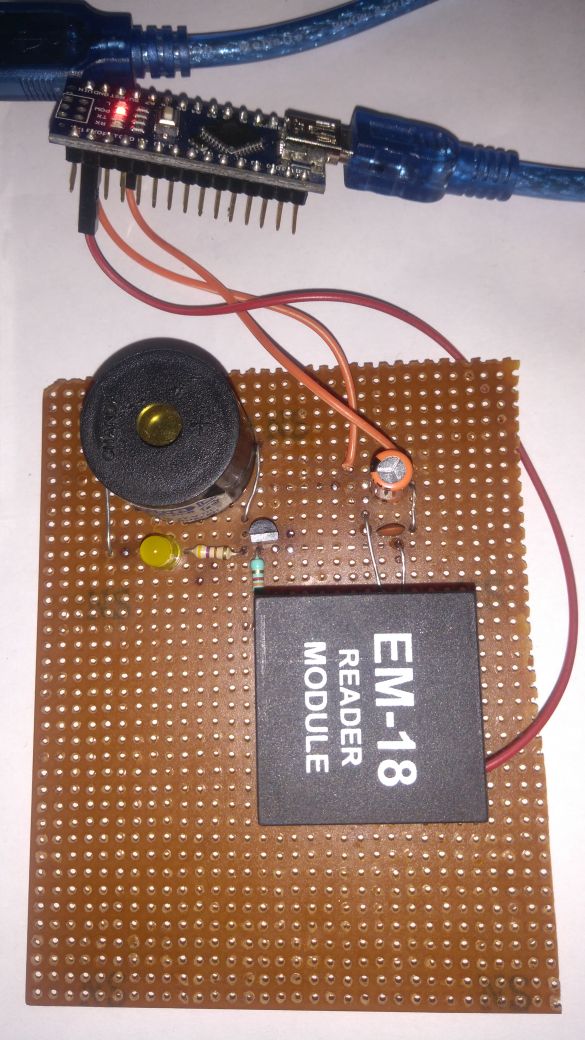


**Fig. 3 RFID Passive tag**

# **RFID reader interfacing circuit**

****

**Fig. 4 RFID reader interfacing circuit**



**Fig. 5 PCB**

# **Code for console application:**

using System;

using System.Collections.Generic;

using System.IO;

using System.IO.Ports;

namespace AttendanceTracker.ConsoleApplication

{

class Program

{

//Dictionary variable to store UID and UName

static Dictionary<string, string> userIdToNameMap;

//Path of attendance log file

const string logFile = "LogFile.txt";

static void Main(string[] args)

{

Console.Title = "Attendance Tracker";

//Populating dictionary variable

userIdToNameMap = GetUserMap();

//File handling - If file doesn't exist create one.

if (!File.Exists(logFile))

{

File.Create(logFile);

}

//Connect to device on serial port

SerialPort serialPort = Connect();

//When data is received(Event) from device OnDataReceived(Event handler method) is called

serialPort.DataReceived += OnDataReceived;

Console.ReadLine();

}

//sender is the (Event triggering) object calling it

private static void OnDataReceived(object sender, SerialDataReceivedEventArgs e)

{

//Typecasting object to SerialPort class and assigning it to serial port variable

var serialPort = (SerialPort)sender;

//Take input from serial port

var uid = serialPort.ReadExisting();

if (uid.Length > 13)

{

// Characters after 13 position might be garbage character and need to be removed.

uid = uid.Substring(0, 12);

}

if (userIdToNameMap.ContainsKey(uid))

{

//Returning UName corresponding to UID(key)

var userName = userIdToNameMap[uid];

//Getting time stamp of event

var timeStamp = DateTime.Now;

//Formatting string

var log = string.Format("{0} logged in @{1}", userName, timeStamp);

//Displaying output on console

Console.WriteLine(log);

//This method creates a StreamWriter object in append mode, on writing to this object data is written to file

var fileWriter = File.AppendText(logFile);

fileWriter.WriteLine(log);

//Close file when writing is complete

fileWriter.Close();

//Dispose of the StreamWriter object

fileWriter.Dispose();

}

}

private static Dictionary<string, string> GetUserMap()

{

Dictionary<string, string> userIdToNameMap = new Dictionary<string, string>();

//Add UID and UName to dictionary, UID is key

userIdToNameMap.Add("18008939F55D", "User A");

userIdToNameMap.Add("5600122FF69D", "User B");

userIdToNameMap.Add("560012456D6C", "User C");

userIdToNameMap.Add("5600122FF398", "User D");

userIdToNameMap.Add("330094EE0148", "User E");

return userIdToNameMap;

}

private static SerialPort Connect()

{

//Connect to COM3 at 9600 baud rate

SerialPort serialPort = new SerialPort(portName: "COM3", baudRate: 9600);

serialPort.Open();

Console.WriteLine("Connected to {0}", serialPort.PortName);

return serialPort;

}

private static void Disconnect(SerialPort serialPort)

{

serialPort.Close();

serialPort.Dispose();

}

}

}

# Code for Arduino application:

//count = 0

int count = 0;

//character array of size 12

char input[12];

//flag =0

boolean flag = 0;

void setup()

{

//begin serial port with baud rate 9600bps

Serial.begin(9600);

}

void loop()

{

if(Serial.available())

{

count = 0;

//Read 12 characters and store them in input array

while(Serial.available() && count < 12)

{

input[count] = Serial.read();

count++;

delay(5);

}

//Print RFID tag number

Serial.println(input);

}

}