

Faculty of Engineering

Mechanical Department

[Mechatronics Program]

# “ Microprocessor & Microcontroller "

CODE: \_ELC252\_\_\_\_\_

PROJECT REPORT

PROJECT: \_\_\_\_SMART HOME\_\_\_\_

GROUP:\_\_\_\_\_\_\_\_\_\_\_\_\_\_



NAME

Omar khaled hassan mohamed ID:2022006780

# Objective:

# The objective of a smart home is to provide home automation and control through the use of interconnected devices that can be managed remotely via a centralized system. This system allows for the automation of various household functions, such as lighting, heating, security, and entertainment, to provide greater convenience, comfort, and energy efficiency for the homeowner. Smart homes can also be integrated with voice assistants and mobile applications for easy and intuitive control. The ultimate goal is to create a more connected, efficient, and personalized living experience for thehomeowner

# Components

## **1.DHT11–Temperature and Humidity Sensor**

### **2.RFID** (Radio-frequency identification

### 3. **HC-5 - Bluetooth Module**

## 4.Buzzer

## 5.Stm32f

## 6.Debuger

## 7. Dc motor

## 8.3leds color

## **1-DHT–Temperature and Humidity Sensor**

[](https://components101.com/sites/default/files/component_pin/DHT11%E2%80%93Temperature-Sensor-Pinout.jpg)

# The **DHT11**is a commonly used **Temperature and humidity sensor that** comes with a dedicated NTC to measure temperature and an 8-bit microcontroller to output the values of temperature and humidity as serial data.

|  |  |  |
| --- | --- | --- |
| **No:** | **Pin Name** | **Description** |
| **DHT11 Sensor module** | | |
| 1 | Vcc | Power supply 3.5V to 5.5V |
| 2 | Data | Outputs both Temperature and Humidity through serial Data |
| 3 | Ground | Connected to the ground of the circuit |

# **DHT11 Specifications**

## Operating Voltage: 3.5V to 5.5V

## Operating current: 0.3mA (measuring) 60uA (standby)

## Output: Serial data

## Temperature Range: 0°C to 50°C

## Humidity Range: 20% to 90%

## Resolution: Temperature and Humidity both are 16-bit

## Accuracy: ±1°C and ±1%

## **2-RFID** (Radio-frequency identification)

# 

|  |  |  |
| --- | --- | --- |
| **Pin Number** | **Pin Name** | **RC522 Pin Configuration** |
| 1 | VCC | Used to supply power (3.3V) to the module. |
| 2 | RST | Used to reset or power down the module. Hard power-down would be activated when this pin goes low. |
| 3 | GND | Ground pin, connected to the GND system of Arduino |
| 4 | IRQ | The interrupt pin is used to alter the [**MCU**](https://www.easybom.com/blog/a/microprocessor-vs-microcontroller-what-is-the-difference) as soon as the RFID tag shows up in its section. |
| 5 | MISO/SCL/Tx | It functions as SCL for the I2C interface and Tx for the UART interface in SPI communication. |
| 6 | MOSI | The Master Out Slave In pin is set to input for SPI communication. |
| 7 | SCK | It provides a clock source for the SPI bus Master (e.g. Arduino). |
| 8 | SS/SDA/Rx | It functions as SS (Serial input) during SPI communication, SDA for IIC, and Rx in UART. |

# **What is RFID Technology?**

## RFID (Radio Frequency Identification) technology is a communication technology that can identify specific targets and read and write relevant data through radio signals without the need to establish mechanical or optical contact between the identification system and specific targets.

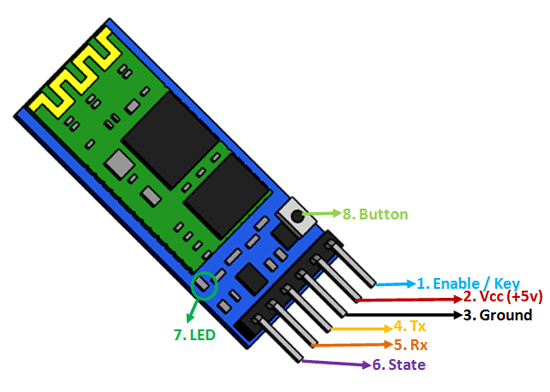
## 

## An RFID system consists of two components: a tag or label and a reader. RFID tags or labels contain both a transmitter and a receiver. The RFID component on the tags is made up of two parts: a microchip that stores and processes data, and an antenna that receives and transmits signals. The tag contains the unique serial number for a single object.

## 

## To read the information encoded on a tag, a two-way radio transmitter-receiver known as an interrogator or reader uses an **antenna** to send a signal to the tag. The tag responds with the data stored in its memory bank. The read results will then be transmitted to an RFID computer program by the interrogator

## RFID working diagram3-**HC-05 - Bluetooth Module**

[](https://components101.com/sites/default/files/component_pin/HC-05-Bluetooth-Module-Pinout.png)

## The HC-05 is a popular bluetooth module which can add two-way (full-duplex) wireless functionality to your projects

|  |  |  |
| --- | --- | --- |
| **Pin Number** | **Pin Name** | **Description** |
| 1 | Enable / Key | This pin is used to toggle between Data Mode (set low) and AT command mode (set high). By default it is in Data mode |
| 2 | Vcc | Powers the module. Connect to +5V Supply voltage |
| 3 | Ground | Ground pin of module, connect to system ground. |
| 4 | TX – Transmitter | Transmits Serial Data. Everything received via Bluetooth will be given out by this pin as serial data. |
| 5 | RX – Receiver | Receive Serial Data. Every serial data given to this pin will be broadcasted via Bluetooth |
| 6 | State | The state pin is connected to on board LED, it can be used as a feedback to check if Bluetooth is working properly. |
| 7 | LED | Indicates the status of Module   * Blink once in 2 sec: Module has entered Command Mode * Repeated Blinking: Waiting for connection in Data Mode * Blink twice in 1 sec: Connection successful in Data Mode |
| 8 | Button | Used to control the Key/Enable pin to toggle between Data and command Mode |

## **HC-05 Default Settings**

## Default Bluetooth Name: “HC-05”

## Default Password: 1234 or 0000

## Default Communication: Slave

## Default Mode: Data Mode

## Data Mode Baud Rate: 9600, 8, N, 1

## Command Mode Baud Rate: 38400, 8, N, 1

## Default firmware: LINVOR

## 

# **HC-05 Technical Specifications**

## Serial Bluetooth module for Arduino  and other microcontrollers

## Operating Voltage: 4V to 6V (Typically +5V)

## Operating Current: 30mA

## Range: <100m

## Works with Serial communication (USART) and TTL compatible

## Follows IEEE 802.15.1 standardized protocol

## Uses Frequency-Hopping Spread spectrum (FHSS)

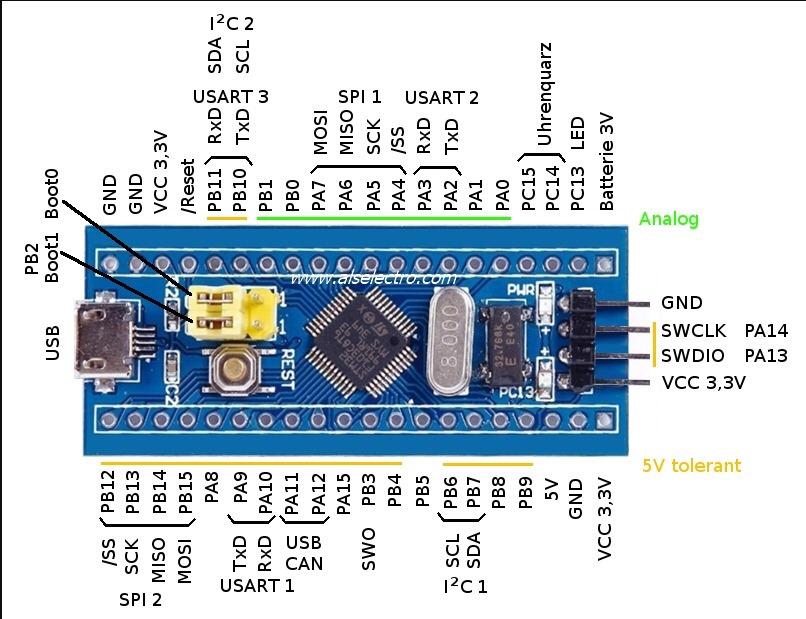
## Can operate in Master, Slave or Master/Slave mode

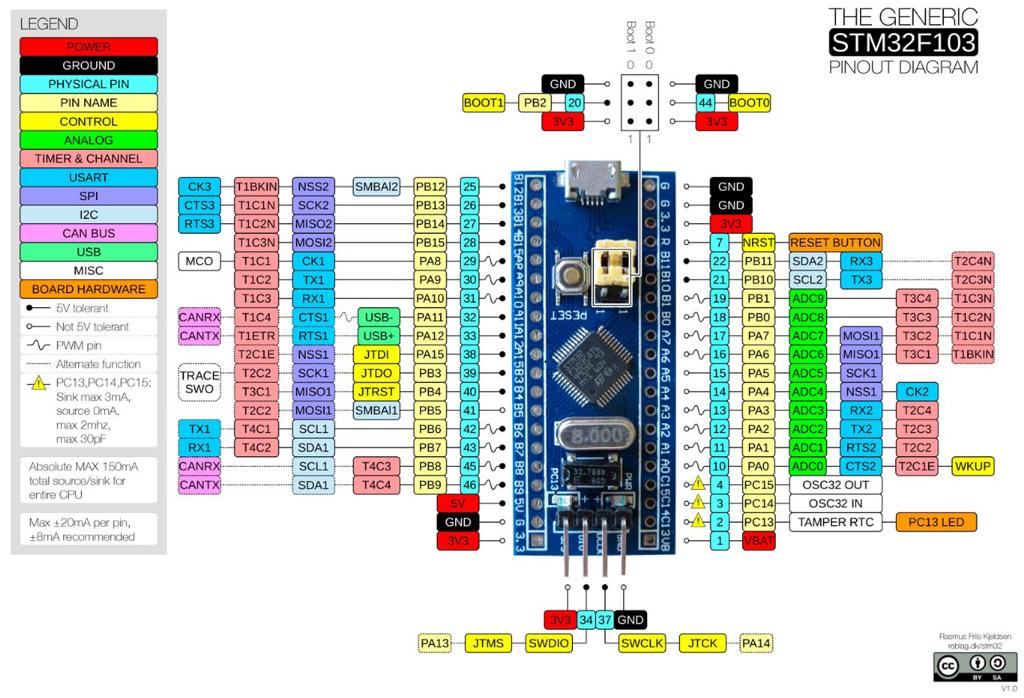
## Can be easily interfaced with Laptop or Mobile phones with Bluetooth

## Supported baud rate: 9600,19200,38400,57600,115200,230400,460800

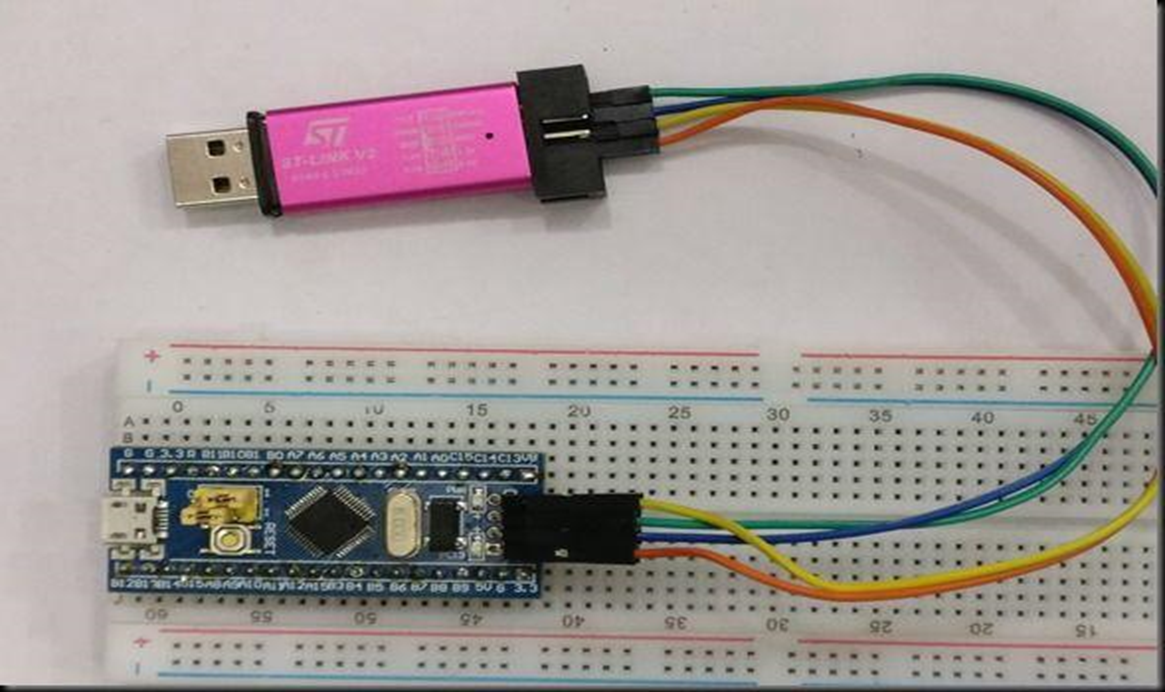
4-Stm32f103b1:

STM32 F1-series was the first group of STM32 microcontrollers based on theARM Cortex-M3 core and considered their mainstream ARM microcontrollers. The F1-series has evolved over time by increasing CPU speed, size of internal memory, variety of peripherals.There are five F1 lines: Connectivity (STM32F105/107), Performance (STM32F103), USB Access (STM32F102), Access (STM32F101), Value (STM32F100

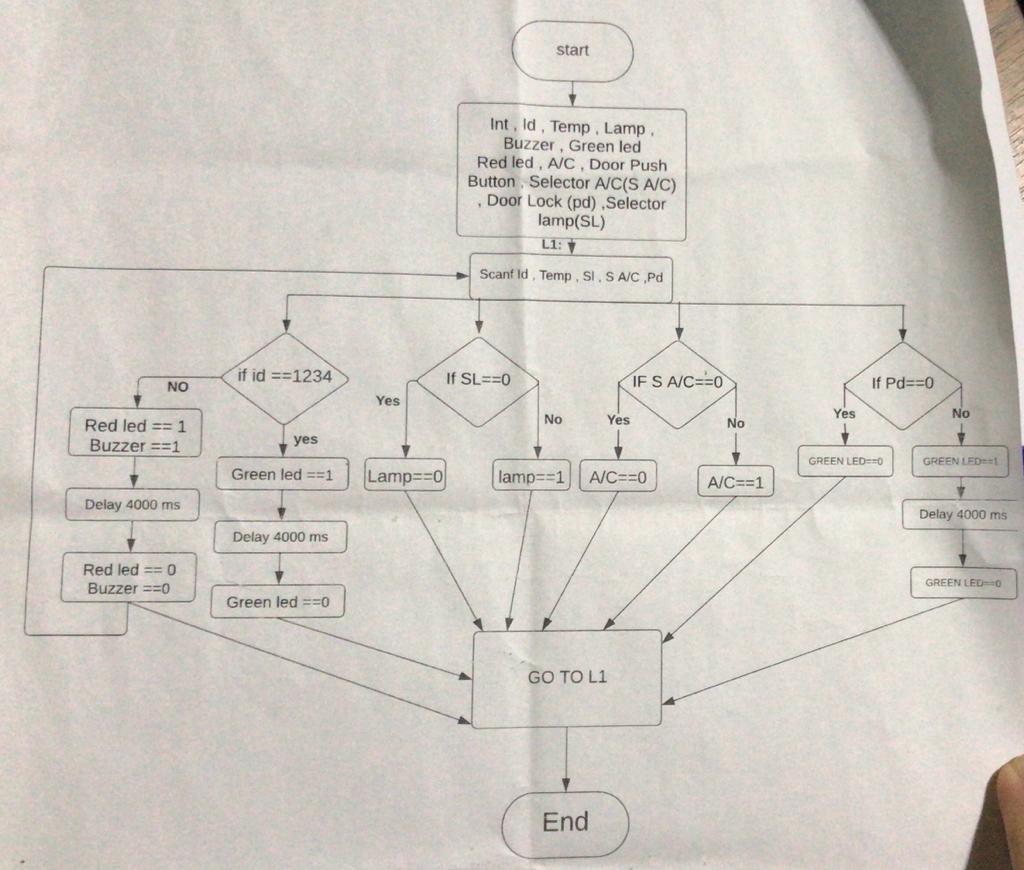


to upload code is through SINGLE WIRE INTERFACE ModuleInterface (SWIM) .This uses only 2 wires SWCLK, SWDIO apart from power pins .You can see this as 4 separate header pins to the right end of module.You need a ST-LINK dongle or device to connect with the SWIM.

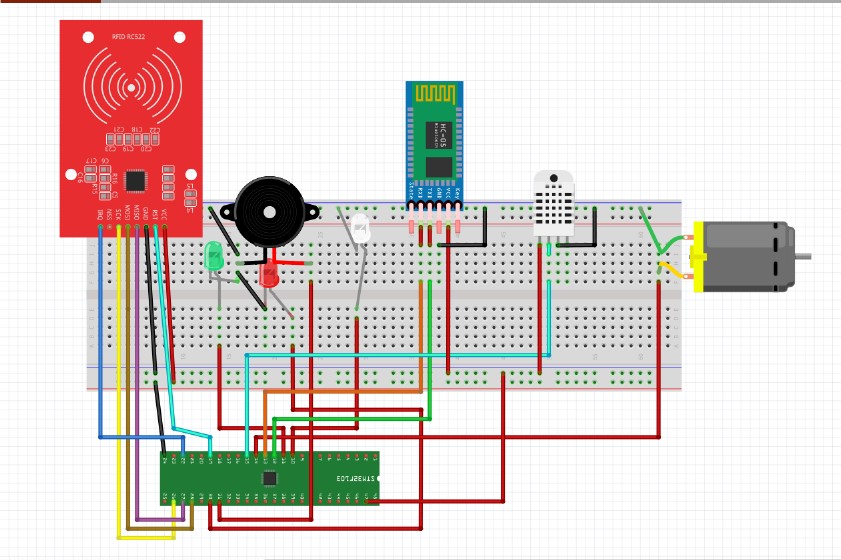
The ST-LINK/V2 is an in-circuit debugger/programmer for the STM8 and STM32 microcontrollers. The single wire interface module (SWIM) interfaces facilitate the communication with any STM8 or STM32 microcontroller operating on an application board



Flow char:



Circuit diagram:



Code –C:

#include <DHT\_U.H>

include <dht.h>

// RemoteXY select connection mode and include library

#define REMOTEXY\_MODE\_\_HARDSERIAL

#include <RemoteXY.h>

#include <DHT.h>

//initilize object dht for class DHT with DHT pin with STM32 and DHT type as DHT11

// RemoteXY connection settings

#define REMOTEXY\_SERIAL Serial2

#define REMOTEXY\_SERIAL\_SPEED 9600

#define DHTPIN PA5

#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

int t;//t=Temp

int h;//h=Humidity

int greenpin ;

int redpin ;

int buzzerpin ;

// RemoteXY configurate

#pragma pack(push, 1)

uint8\_t RemoteXY\_CONF[] = // 139 bytes

{ 255,3,0,22,0,132,0,16,31,1,2,0,24,7,22,11,12,26,31,31,

79,78,0,79,70,70,0,67,4,7,60,20,5,2,26,11,129,0,8,51,

18,6,17,84,101,109,112,0,129,0,2,33,26,6,17,108,111,99,107,32,

100,111,111,114,0,129,0,3,11,18,6,17,76,97,109,112,0,2,0,10,

76,22,11,12,26,31,31,79,78,0,79,70,70,0,129,0,12,68,18,6,

17,65,47,67,0,1,8,33,29,12,12,12,27,0,67,4,33,60,20,5,

2,26,11,129,0,33,50,18,6,17,104,117,109,105,100,105,116,121,0 };

// this structure defines all the variables and events of your control interface

struct {

// input variables

uint8\_t lamp; // =1 if switch ON and =0 if OFF

uint8\_t air\_condition; // =1 if switch ON and =0 if OFF

uint8\_t lock\_door; // =1 if button pressed, else =0

// output variables

char text\_Temp[11]; // string UTF8 end zero

char text\_Humidity[11]; // string UTF8 end zero

// other variable

uint8\_t connect\_flag; // =1 if wire connected, else =0

} RemoteXY;

#pragma pack(pop)

//////switchs

#define PIN\_LAMP PA0

#define PIN\_AIR\_CONDITION PA4

#define PIN\_LOCK\_DOOR PA1

///////DHT

//////////RFID

#include <SPI.h>

#include <MFRC522.h>

#define SS\_PIN PB11

#define RST\_PIN PB1

MFRC522 rfid(SS\_PIN, RST\_PIN); // Instance of the class

MFRC522::MIFARE\_Key key;

// Init array that will store new NUID

byte nuidPICC[4];

#define greenpin PA1

#define redpin PA9

#define buzzerpin PA10

void setup()

{

////////DHT

Serial.begin(9600);

display.begin(SSD1306\_SWITCHCAPVCC, 0x3C); // initialize with the

pinMode(DHTPIN, OUTPUT);

dht.begin();

/////

RemoteXY\_Init ();

pinMode (PIN\_LAMP, OUTPUT);

pinMode (PIN\_AIR\_CONDITION, OUTPUT);

pinMode (PIN\_LOCK\_DOOR, OUTPUT);

////ID locker

pinMode(greenpin, OUTPUT);

pinMode(redpin, OUTPUT);

pinMode(buzzerpin, OUTPUT);

Serial.begin(9600);

SPI.begin(); // Init SPI bus

rfid.PCD\_Init(); // Init MFRC522

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

key.keyByte[i] = 0xFF;

}

Serial.println(F("This code scan the MIFARE Classsic NUID."));

Serial.print(F("Using the following key:"));

printHex(key.keyByte, MFRC522::MF\_KEY\_SIZE);

}

// TODO you setup code

void loop()

{

RemoteXY\_Handler ();

////switchs

digitalWrite(PIN\_LAMP, (RemoteXY.lamp==0)?LOW:HIGH);

digitalWrite(PIN\_AIR\_CONDITION, (RemoteXY.air\_condition==0)?LOW:HIGH);

digitalWrite(PIN\_LOCK\_DOOR, (RemoteXY.lock\_door==0)?LOW:HIGH);

///////////DHT

h = dht.readHumidity(); //Gets Humidity value

t = dht.readTemperature(); //Gets Temperature value

dtostrf(t, 0, 1, RemoteXY.text\_Temp);

dtostrf(h, 0, 1, RemoteXY.text\_Humidity);

delay(1500);

///////ID LOcker

// Reset the loop if no new card present on the sensor/reader. This saves the entire process when idle.

if ( ! rfid.PICC\_IsNewCardPresent())

return;

// Verify if the NUID has been readed

if ( ! rfid.PICC\_ReadCardSerial())

return;

Serial.print(F("PICC type: "));

MFRC522::PICC\_Type piccType = rfid.PICC\_GetType(rfid.uid.sak);

Serial.println(rfid.PICC\_GetTypeName(piccType));

// Check is the PICC of Classic MIFARE type

if (piccType != MFRC522::PICC\_TYPE\_MIFARE\_MINI &&

piccType != MFRC522::PICC\_TYPE\_MIFARE\_1K &&

piccType != MFRC522::PICC\_TYPE\_MIFARE\_4K) {

Serial.println(F("Your tag is not of type MIFARE Classic."));

return;

}

if (rfid.uid.uidByte[0] != nuidPICC[0] ||

rfid.uid.uidByte[1] != nuidPICC[1] ||

rfid.uid.uidByte[2] != nuidPICC[2] ||

rfid.uid.uidByte[3] != nuidPICC[3] ) {

Serial.println(F("A new card has been detected."));

// Store NUID into nuidPICC array

for (byte i = 0; i < 4; i++) {

nuidPICC[i] = rfid.uid.uidByte[i];

}

Serial.println(F("The NUID tag is:"));

Serial.print(F("In hex: "));

printHex(rfid.uid.uidByte, rfid.uid.size);

Serial.println();

Serial.print(F("In dec: "));

printDec(rfid.uid.uidByte, rfid.uid.size);

Serial.println();

}

else Serial.println(F("Card read previously."));

if(nuidPICC[0] == 0x73 && nuidPICC[1]==0xBE && nuidPICC[2]==0xFC && nuidPICC[3]==0x00){

digitalWrite(greenpin , HIGH);

delay(4000);

digitalWrite(greenpin , LOW);

}

else{

digitalWrite(redpin,HIGH);

digitalWrite(buzzerpin, HIGH);

delay(4000);

digitalWrite(redpin,LOW);

digitalWrite(buzzerpin,LOW);

}

// Halt PICC

rfid.PICC\_HaltA();

// Stop encryption on PCD

rfid.PCD\_StopCrypto1();

}

// TODO you loop code

// use the RemoteXY structure for data transfer

// do not call delay(), use instead RemoteXY\_delay()

void printHex(byte \*buffer, byte bufferSize) {

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

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

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

}

}

/\*\*

\* Helper routine to dump a byte array as dec values to Serial.

\*/

void printDec(byte \*buffer, byte bufferSize) {

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

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

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

}

}

Code\_assembly:

; Include libraries

#include <DHT.h>

#include <RemoteXY.h>

#include <SPI.h>

#include <MFRC522.h>

; Define constants

#define DHTPIN PA5

#define DHTTYPE DHT11

#define REMOTEXY\_SERIAL Serial2

#define REMOTEXY\_SERIAL\_SPEED 9600

#define PIN\_LAMP PA0

#define PIN\_AIR\_CONDITION PA4

#define PIN\_LOCK\_DOOR PA1

#define SS\_PIN PB11

#define RST\_PIN PB1

#define greenpin PA1

#define redpin PA9

#define buzzerpin PA10

; Initialize DHT object

DHT dht(DHTPIN, DHTTYPE);

; Define variables

int t; ; Temperature

int h; ; Humidity

int greenpin;

int redpin;

int buzzerpin;

; RemoteXY configuration

#pragma pack(push, 1)

uint8\_t RemoteXY\_CONF[] = ; 139 bytes

{ 255,3,0,22,0,132,0,16,31,1,2,0,24,7,22,11,12,26,31,31,

79,78,0,79,70,70,0,67,4,7,60,20,5,2,26,11,129,0,8,51,

18,6,17,84,101,109,112,0,129,0,2,33,26,6,17,108,111,99,107,32,

100,111,111,114,0,129,0,3,11,18,6,17,76,97,109,112,0,2,0,10,

76,22,11,12,26,31,31,79,78,0,79,70,70,0,129,0,12,68,18,6,

17,65,47,67,0,1,8,33,29,12,12,12,27,0,67,4,33,60,20,5,

2,26,11,129,0,33,50,18,6,17,104,117,109,105,100,105,116,121,0 };

; Define control interface structure

struct {

; Input variables

uint8\_t lamp; ; =1 if switch ON and =0 if OFF

uint8\_t air\_condition; ; =1 if switch ON and =0 if OFF

uint8\_t lock\_door; ; =1 if button pressed, else =0

; Output variables

char text\_Temp[11]; ; string UTF8 end zero

char text\_Humidity[11]; ; string UTF8 end zero

; Other variable

uint8\_t connect\_flag; ; =1 if wire connected, else =0

} RemoteXY;

#pragma pack(pop)

; Initialize MFRC522 object

MFRC522 rfid(SS\_PIN, RST\_PIN); ; Instance of the class

MFRC522::MIFARE\_Key key;

; Init array that will store new NUID

byte nuidPICC[4];

; Main program

void setup() {

; Initialize DHT sensor

dht.begin();

; Initialize RemoteXY

RemoteXY\_Init();

; Initialize SPI communication

SPI.begin();

; Initialize MFRC522

rfid.PCD\_Init();

}

void loop() {

; RemoteXY control

RemoteXY\_Handler();

; Read DHT sensor

t = dht.readTemperature();

h = dht.readHumidity();

; Update RemoteXY variables

RemoteXY.text\_Temp = t;

RemoteXY.text\_Humidity = h;

; Check RFID card

if (rfid.PICC\_IsNewCardPresent() && rfid.PICC\_ReadCardSerial()) {

; Read card UID

for (byte i = 0; i < 4; i++) {

nuidPICC[i] = rfid.uid.uidByte[i];

}

}

}

; Assembly code for Arduino sketch

; Define constants

DHTPIN PA5

PIN\_LAMP PA0

PIN\_AIR\_CONDITION PA4

PIN\_LOCK\_DOOR PA1

greenpin PA1

redpin PA9

buzzerpin PA10

; Initialize setup

setup:

call Serial\_begin

call display\_begin

mov r16, DHTPIN

out DDRB, r16 ; Set DHTPIN as output

call dht\_begin

call RemoteXY\_Init

mov r16, PIN\_LAMP

out DDRB, r16 ; Set PIN\_LAMP as output

mov r16, PIN\_AIR\_CONDITION

out DDRB, r16 ; Set PIN\_AIR\_CONDITION as output

mov r16, PIN\_LOCK\_DOOR

out DDRB, r16 ; Set PIN\_LOCK\_DOOR as output

mov r16, greenpin

out DDRB, r16 ; Set greenpin as output

mov r16, redpin

out DDRB, r16 ; Set redpin as output

mov r16, buzzerpin

out DDRB, r16 ; Set buzzerpin as output

call Serial\_begin

call SPI\_begin

call PCD\_Init

ldi r16, 0xFF

ldi r17, 0

init\_loop:

st Z+, r16 ; Initialize key.keyByte with 0xFF

inc r17

cpi r17, 6

brne init\_loop

call print\_key\_info

ret

; Main loop

loop:

call RemoteXY\_Handler

lds r16, RemoteXY\_lamp

cpi r16, 0

breq lamp\_off

rjmp lamp\_on

lamp\_off:

ldi r16, 0

out PORTB, r16 ; Turn off PIN\_LAMP

rjmp air\_condition\_check

lamp\_on:

ldi r16, 1

out PORTB, r16 ; Turn on PIN\_LAMP

air\_condition\_check:

lds r16, RemoteXY\_air\_condition

cpi r16, 0

breq air\_condition\_off

rjmp air\_condition\_on

air\_condition\_off:

ldi r16, 0

out PORTB, r16 ; Turn off PIN\_AIR\_CONDITION

rjmp lock\_door\_check

air\_condition\_on:

ldi r16, 1

out PORTB, r16 ; Turn on PIN\_AIR\_CONDITION

lock\_door\_check:

lds r16, RemoteXY\_lock\_door

cpi r16, 0

breq lock\_door\_off

rjmp lock\_door\_on

lock\_door\_off:

ldi r16, 0

out PORTB, r16 ; Turn off PIN\_LOCK\_DOOR

rjmp read\_dht

lock\_door\_on:

ldi r16, 1

out PORTB, r16 ; Turn on PIN\_LOCK\_DOOR

read\_dht:

call dht\_readHumidity

call dht\_readTemperature

call dtostrf\_t

call dtostrf\_h

call RemoteXY\_delay

rjmp id\_locker

; ID Locker

id\_locker:

call PICC\_IsNewCardPresent

breq id\_locker\_end

call PICC\_ReadCardSerial

breq id\_locker\_end

call print\_PICC\_type

call check\_PICC\_type

call check\_nuid

rjmp id\_locker\_end

id\_locker\_end:

call PICC\_HaltA

call PCD\_StopCrypto1

ret

; Helper functions

Serial\_begin:

; Code for Serial.begin(9600)

ret

display\_begin:

; Code for display.begin(SSD1306\_SWITCHCAPVCC, 0x3C)

ret

dht\_begin:

; Code for dht.begin()

ret

RemoteXY\_Init:

; Code for RemoteXY\_Init()

ret

SPI\_begin:

; Code for SPI.begin()

ret

PCD\_Init:

; Code for rfid.PCD\_Init()

ret

print\_key\_info:

; Code for printing key information

ret

dht\_readHumidity:

; Code for dht.readHumidity()

ret

dht\_readTemperature:

; Code for dht.readTemperature()

ret

dtostrf\_t:

; Code for converting temperature to string

ret

dtostrf\_h:

; Code for converting humidity to string

ret

RemoteXY\_delay:

; Code for RemoteXY\_delay()

ret

PICC\_IsNewCardPresent:

; Code for rfid.PICC\_IsNewCardPresent()

ret

PICC\_ReadCardSerial:

; Code for rfid.PICC\_ReadCardSerial()

ret

print\_PICC\_type:

; Code for printing PICC type

ret

check\_PICC\_type:

; Code for checking PICC type

ret

check\_nuid:

; Code for checking nuid

ret

PICC\_HaltA:

; Code for rfid.PICC\_HaltA()

ret

PCD\_StopCrypto1:

; Code for rfid.PCD\_StopCrypto1()

ret

Refrance:

RFID :<https://www.easybom.com/blog/a/rc522-rfid-module-arduino-module-pin-configurations-tutorial?gclid=EAIaIQobChMIvZeOpKmcgwMV-J-DBx2vUgDAEAAYAiAAEgKKBfD_BwE>

DHT: https://components101.com/sensors/dht11-temperature-sensor

HC-05: https://components101.com/wireless/hc-05-bluetooth-module