

GAZIANTEP UNIVERSITY

**WIRELESS MINI WEATHER STATION**

**EEE 499 GRADUATION PROJECT**

**IN**

**ELECTRICAL & ELECTRONICS ENGINEERING**

**BY NURGÜL İBİLİ**

**JUNE 2020**

# ABSTRACT

**Wireless Mini Weather Station**

In this project,wireless mini weather station construction is explained.In this arduino based project,the parameters such as pressure,altitude,temperature and humidity taken from the sensors were taken from the transmitter part via wireless communication and transferred to the LCD screen in the receiver part.In this way,by placing the transmitter part in the open area you want the meteorological changes can be monitored on the LCD screen,as well as the graphical change can be observed with the interface.

**Keywords:** Wireless Communication, Weather Station, LCD,Graphics

# ÖZET

**Kablosuz Mini Hava Durumu İstasyonu**

**Mezuniyet Projesi, Elektrik ve Elektronik Mühendisliği Bölümü**

**Haziran 2020, 34 sayfa**

.

Bu projede Kablosuz Mini Hava Durumu İstasyonu yapımı açıklanmıştır.Arduino tabanlı bu proje de verici kısımda bulunan sensörlerden alınan veriler kablosuz haberleşme yoluyla LCD ekrana aktarılmıştır.Bu şekilde verici kısım herhangi bir alana yerleştirilerek meteorolojik değişiklikler LCD ekranda görüntülenebilindiği gibi arayüz yardımıyla grafiksel değişimide gözlemlenebilmektedir.

**Anahtar kelimeler:** Kablosuz Haberleşme, Hava Durumu İstasyonu,LCD

**TABLE OF CONTENTS**

**Page**

[ABSTRACT ii](#_TOC_250016)

[ÖZET iii](#_TOC_250015)

ACKNOWLEDGEMENT 1

LIST OF FIGURES 2-3

LIST OF SYMBOLS 4

1. [INTRODUCTION 5](#_TOC_250014)
2. [HARDWARE 7](#_TOC_250013)
   1. [TEMPERATURE AND HUMIDITY MODULE… **7**](#_TOC_250012)

[2.2. PRESSURE MODULE **8**](#_TOC_250011)

* 1. [TIME MODULE **9**](#_TOC_250010)
  2. [LIQUID CRYSTAL DISPLAY(LCD) **10**](#_TOC_250009)
  3. [ARDUINO UNO **11**](#_TOC_250008)
  4. NRFL01 MODULE **13**
  5. [BATTERY **14**](#_TOC_250007)

1. [DESIGN of THE WIRELESS MINI WEATHER STATION 16-21](#_TOC_250006)
2. [SOFTWARE 22-30](#_TOC_250005)
3. [CONCLUSION 31](#_TOC_250004)

[REFERENCES 32](#_TOC_250003)

[APPENDICES](#_TOC_250002)

1. [COST ANALYSIS 33](#_TOC_250001)
2. [GANTT CHART 34](#_TOC_250000)

**LIST OF FIGURES**

**Page**

Figure 2.1 DHT11 Module 7

Figure 2.1.1 DHT11-MCU Connections 7

Figure 2.2 BMP180 Module 8

Figure 2.2.1 BMP180-MCU Connections 8

Figure 2.3 DS3231 Module 9

Figure 2.3.1 DS3231-MCU Connections… 9

Figure 2.4 LCD(4\*20) 10

Figure 2.4.1 LCD-MCU Connections 10

Figure 2.5 Arduino Uno 11

Figure 2.5.1 Arduino Uno Pin Mapping 12

Figure 2.6 NRF24L01 Module 13

Figure 2.6.1 NRF24L01-MCU Connections 13

Figure 2.7 Battery 14

Figure 3.1.1 Transmitter Part 18

Figure 3.1.2 Receiver Part 19

Figure 3.2.1 Interface Design 20

Figure 3.2.2 Graphic Comment 21

Figure 5.1 Results on LCD 31

**LIST OF SYMBOLS**

**°C Degrees Celsius**

**% Percantage**

**m meter**

**Pa Pascal**

**hPa hectopascal**

**MCU Microcontroller unit**

# INTRODUCTION

The general purpose of our project is to design a wireless mini weather station. Unlike normal weather stations, we have added pressure, altitude and time zone. We need performance metrics for the materials we use to achieve the project’s goals. We took care to use quality products to make or measurements more accurate. We worked on the electronic properties of the materials which are used for in our projects. On the other hand, proper spelling codes for the electronic part are arranged. The software is based on the Arduino platform. In addition, we have the opportunity to graphically examine the interface with the C# program we observe the data coming to the weather station. In addition, our project consists of two pieces as transmitter and receiver. We perform data transfer thanks to through wireless.

Weather forecasting stations are systems that allow forecasting of daily, weekly or monthly weather. These systems, which are used by meteorology in our country, can be both difficult and costly for individual use. Smart weather stations are being developed that can be used individually in order to get rid of such problems. In this study, a smart weather station has realized for the monitoring of weather conditions when changing during the day. The data that received from the temperature, humidity and pressure in the air station, are processed by an Arduino-based processor and then estimated weather information has been given to users. The study results have been compared with results obtained from meteorology and the results have been seen to be close to each other.

# HARDWARE

# TEMPERATURE AND HUMIDITY MODULE

DHT11 is a digital temperature sensor with its own processor. (8 bit). The plus of this sensor is that it gives us ambient humidity as well as temperature. It is an ideal temperature sensor.

* + - The unit measures temperature between 0 and 50 ° C with a margin of 2° C and measures humidity between 20-90% with 5% RH margin.
    - It is highly reliable and stable in long-term studies.

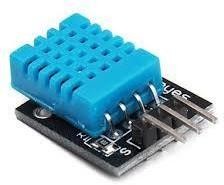


Figure2.1 DHT11 Module

# 2.1.1. DHT11 MODULE-MCU CONNECTION

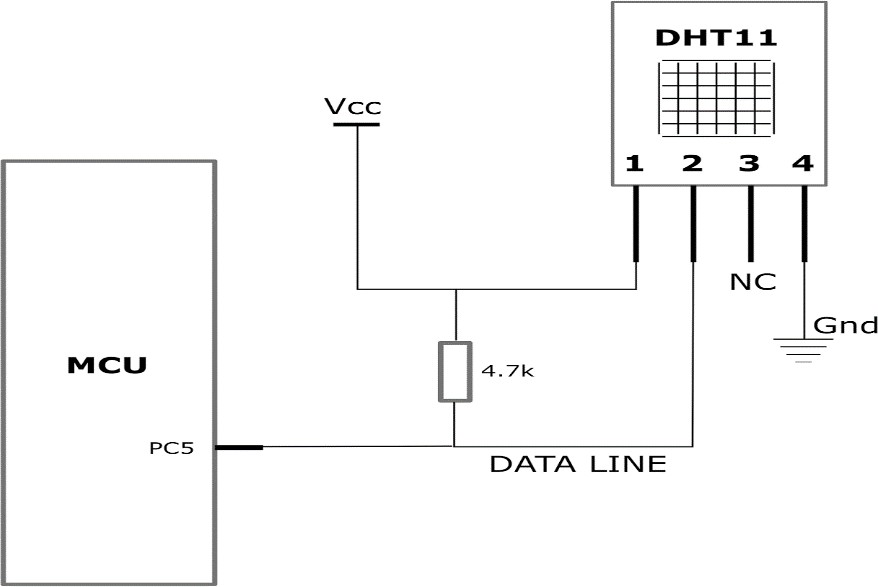


Figure2.1.1 DHT11-MCU Connections

# PRESSURE MODULE

The BMP180 pressure sensor is a product with a very small size, which digitally outputs by measuring the air pressure in the environment. Due to the understanding of the sensor height according to the air pressure, it can be used in many systems, especially in aircrafts projects. It gives Information about the altitude between 500-9000 meters. It has a very high resolution such as 0.03 h Pa (0.25 meters). It can also measure the temperature in the environments.



Figure2.2 BMP180 Module

# 2.2.1)BMP180 MODULE-MCU CONNECTION

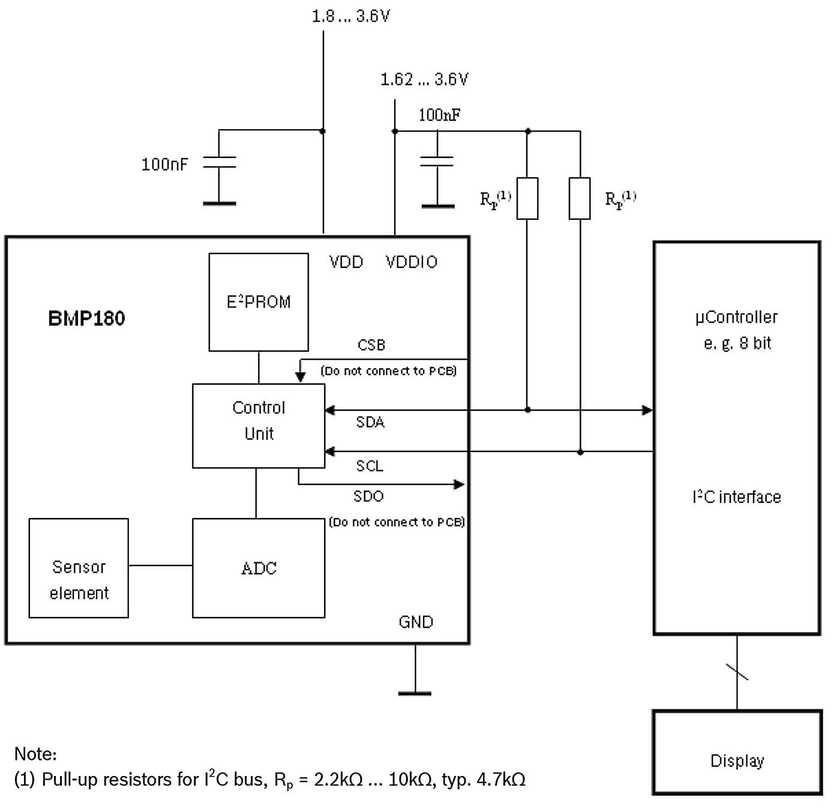


Figure2.2.1 BMP180-MCU Connections

# TIME MODULE

As we know, we can see that the DS3231 RTC module is a little higher quality and sensitive than other RTC modules, meaning that it counts time more consistently. It can work for a long time without time deviation. The beauty of this RTC module is that it has an interval temperature sensor on it. the main purpose of this sensor is to balance frequency distortions due to temperature changes and show more accurate time. But if we wish this temperature sensor, we can use it as a sensor that measures the temperature in our project. For example, if we practice with a watch and

a degree, we will no need to buy individual sensors for both we watch and temperature.



Figure2.3 DS3231 Module

# 2.3.1. DS3231 MODULE-MCU CONNECTION

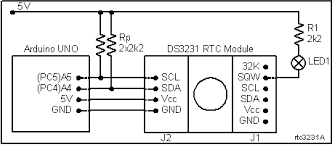


Figure2.3.1 DS3231-MCU Connections

# LIQUID CRYSTAL DISPLAY(LCD)

Liquid Crystal Display that is electrically polarized light is a liquid display technology based

on single-phase goals and added into a Polarization filter can be seen with the eye principle. Liquid crystals in the LCD’s can be found in thermotropic and lyotropic phases depending on the temperature and substance structure. The varieties of noematic liquid crystal, which are subset of thermotropic phase liquid crystal, called twisted noematic (TN), become flattie. non-folded noematic, depending on the voltage of the applied current. Noematic liquid crystals are the liquid crystal phase that makes it possible to make LCDs. In order to make LCDs, the light should be polarized, the liquid crystals should be able to pass polarized light, the molecular arrangement of the liquid crystals should be changed by electric current and a structure that conducts electricity.

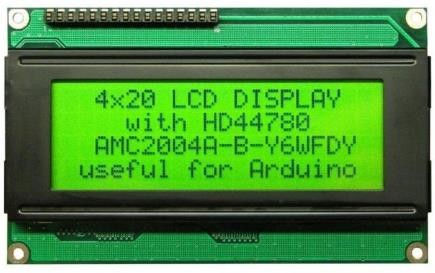


Figure2.4 LCD( 4\*20)

# 2.4.1. LCD-MCU CONNECTION

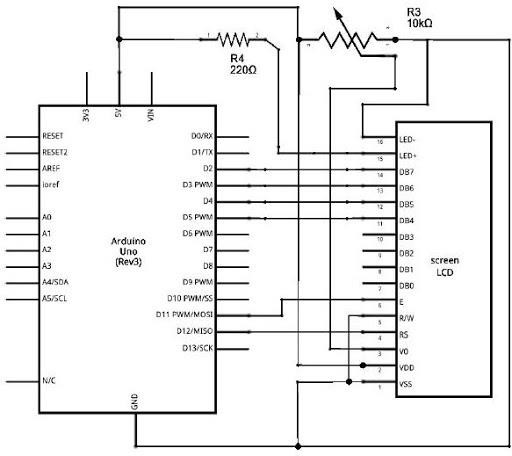


Figure2.4.1 LCD-MCU Connections

# ARDUINO UNO

With Arduino Uno, you can get physical information from various sensors, and you can do various experiments with this information. You can also get an output from stimuli such as motor, LED, buzzer. A basic programming knowledge is sufficient to control such electronic components by connecting them to the Arduino Uno board. The level of electronics and programming knowledge required will increase according to the level of the projects. Although there are much smaller and bigger models in size, the size of Arduino Uno is the most standard according to the projects. Having 14 digital output means that 14 different digital sensors and exciter can be controlled. This is a sufficient number for manyprojects.5 of these digital outputs are PWM outputs. The stimuli that

are desired to be controlled analogously, such as the speed of the motors and the brightness levels in the LEDs, are controlled by connecting to these PWM pins.6 analog inputs in Arduino Uno are for the sensors that we can receive analog input signal. With Arduino Uno, you can also make more advanced projects such as drone, robot, smart home automation, burglar system from the basic applications such as LED flashing. This is all about what you want to do. In short, Arduino Uno

is a standard sized control board that allows you to control electronic circuits in many applications from simple to difficult.



Figure2.5 Arduino Uno

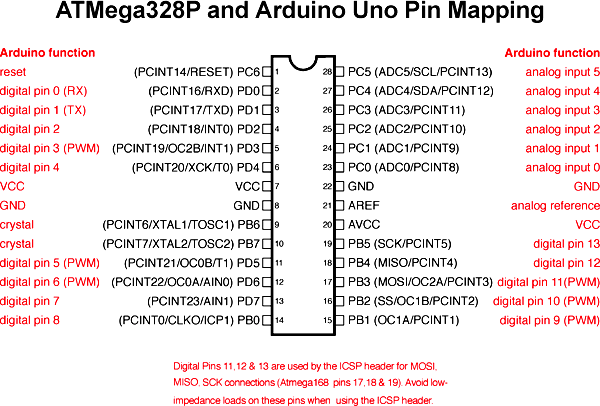


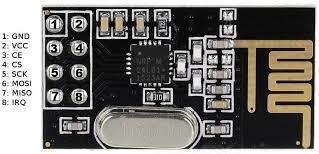
Figure2.5.1 Arduino Uno Pin Mapping

# WIRELESS MODULE (NRF24L01)

It is a digital radio frequency wireless communication chip developed by order company that works both in receiver and transmitter feature in the 2.4 Ghz frequency band. Below is the image of this chip NRF24L01 communicates with microcontrollers with SPI communication protocol.

Communication distance of this module is 250m in open area. Transmitter signal power is

+7 dB and receiver sensitivity are less than 90dB.

Figure2.6 NRF24L01 Module

# NRF24L01 MODULE-MCU CONNECTION

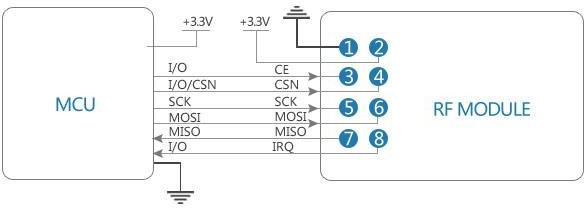


Figure2.6.1 NRF24L01 -MCU Connections

# 2.7. BATTERY

The battery is an electronic device that converts chemical energy into electrical energy. It has two terminals as anode (-tip) and cathode (+tip). The tip marked as minus is the source of electrons and acts as a terminal using electrons connected to a device that uses electrical energy. The ions of the motion electrons move and perform chemical reaction to generate easy. The word “battery “is the name given to the structure consisting of multiple cells (we can call

a battery) where these reactions take place separately from each other. As technology evolves, higher energy production per cell becomes possible without the need for a larger number of cells.



Figure2.7 Battery

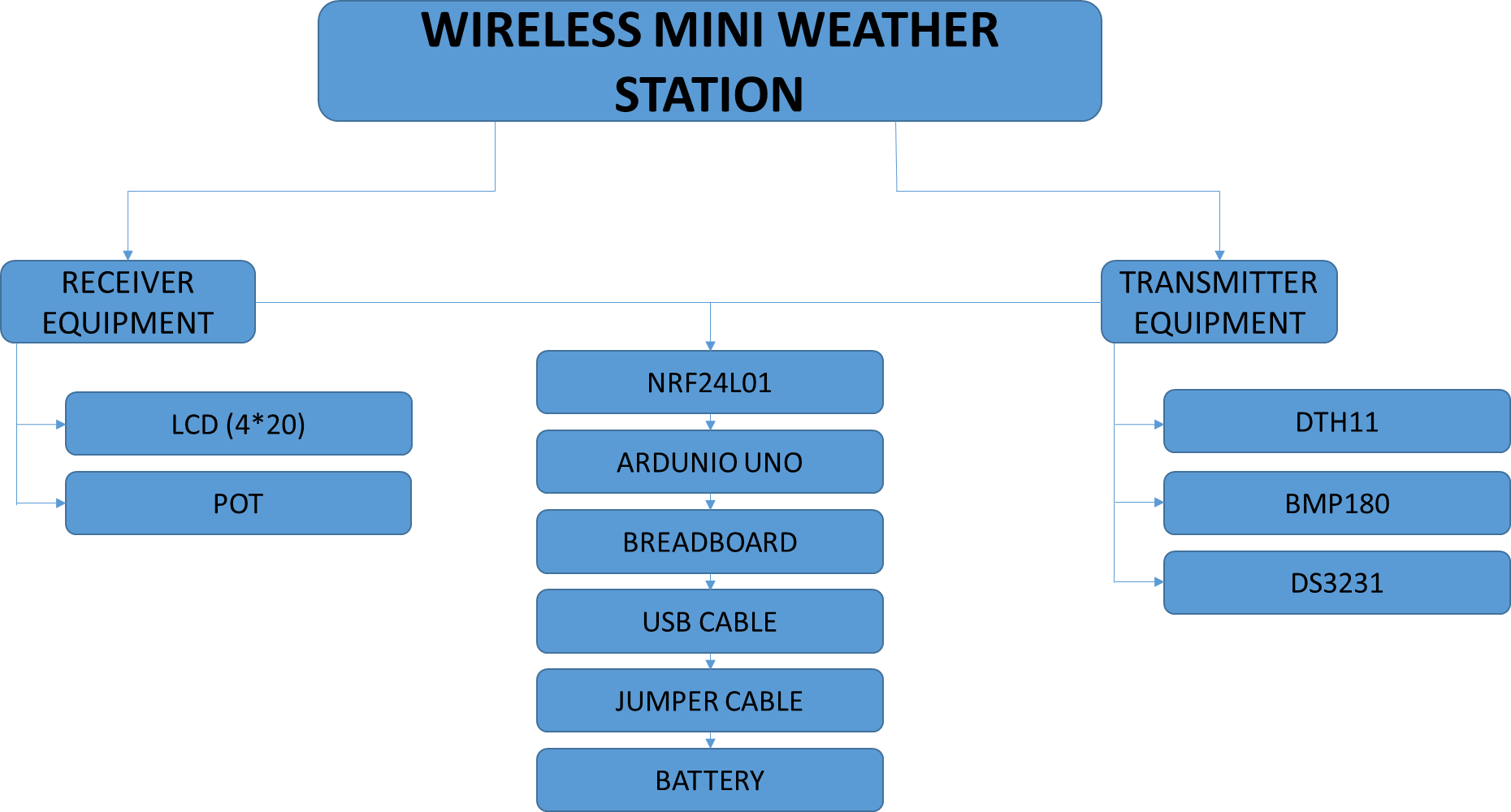


Table1 Wireless Mini Weather Station

# DESIGN of THE WIRELESS MINI WEATHER STATION

We used the fritzing program to realize the design. We place the sensors on the circuit board in the appropriate empty places and then connect the appropriate jumper cables of Arduino and then our circuit is ready to use. One important thing is that we need to test each sensor to see if itworks.

# MECHANICAL DESIGN

* + 1. **TRANSMITTER PART DESIGN**
    2. **RECEIVER PART DESIGN**

# C# INTERFACE DESIGN

* 1. **MECHANICAL PART**

In this section we show the connection shapes of the materials used. Transmitter and receiver part connection. We used the fritzing program to realize the design. We put the sensors in the empty spaces suitable for the circuit board, connect the appropriate jumper cables of Arduino, and then the circuit is ready for use. One important thing is that we need to test whether each sensor is working. In the transmitter part, it worked by taking the Arduino power from the battery. We used our sensors in this section. We used one of the NRF24l01 module in this section to provide wireless communication. To transfer data to the receiver. The receiver has worked by taking the Arduino power from the battery. We installed the NRF24L01 module to provide wireless communication to the receiver. We transferred the results from the sensors in the transmitter part to the receiver part. And we used one LCD screen. We showed the results on the LCD screen.

# TRANSMITTER PART

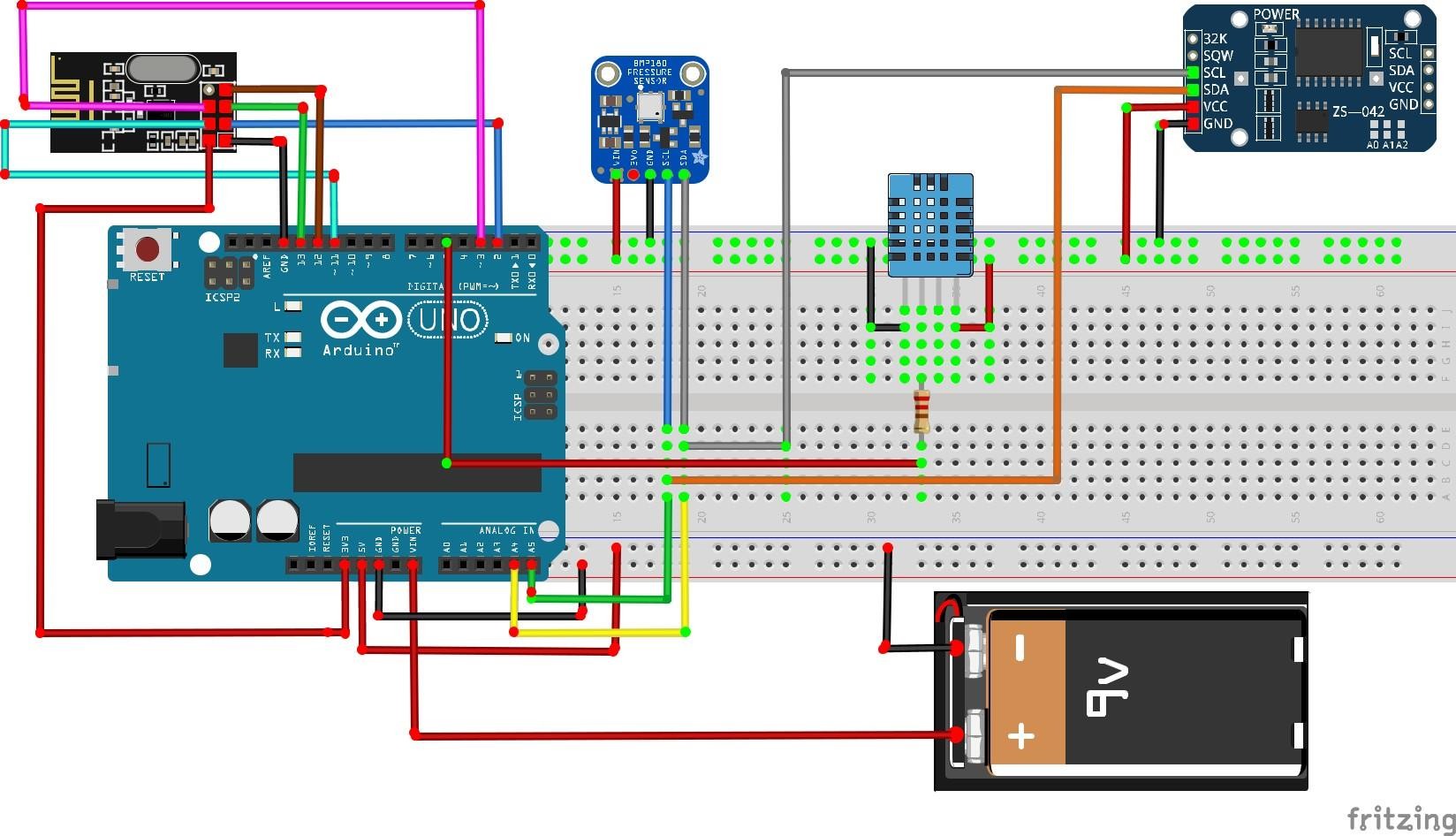


Figure3.1.1 Transmitter Part

We used our sensors in this section.Above the figure,from left to right there are wireless communication,Pressure and altitude sensor,temperature and humidity sensor and real

time module.In the transmitter part,it worked by taking the Arduino power from the battery. We used one of the NRF24L01 module in this section to provide wireless communication to transfer data to receiver.

# RECEIVER PART

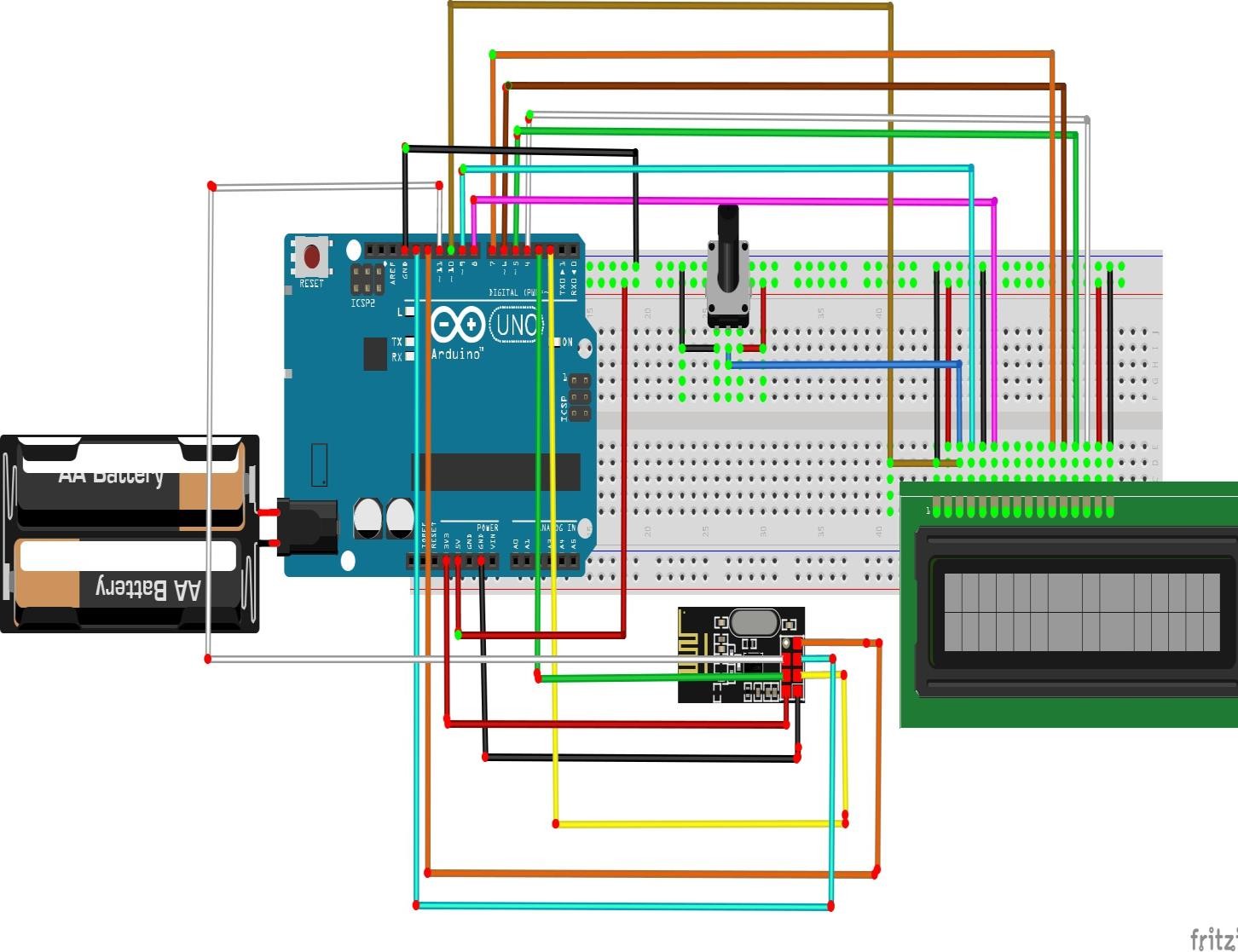


Figure3.1.2 Receiver Part

The receiver has worked by taking the Arduino power from the battery. We installed the NRF24L01 module to provide wireless communication to the receiver. We transferred the results from the sensors in the transmitter part to the receiver part. And we used one LCD screen. We showed the results on the LCD screen.

# C# INTERFACE DESIGN

In the interface design we have used Chart**.** The best way to keep track of incoming data for this project would be to use graphics. Using the graph, we can easily interpret meteorological data that changes throughout the day.

* There are 4 different lines in this chart.
* Brown line shows temperature values,blue line shows humidity values,orange line shows pressure values and red altitude values.
* We observe the change of lines over time.
* We can see the maximum points of the lines.
* The Y-Axis shows the first y-axis temperature and humidity, and the second y-axis altitude and pressure.
* X Axis shows time change.



Figure3.2.1 Interface Design

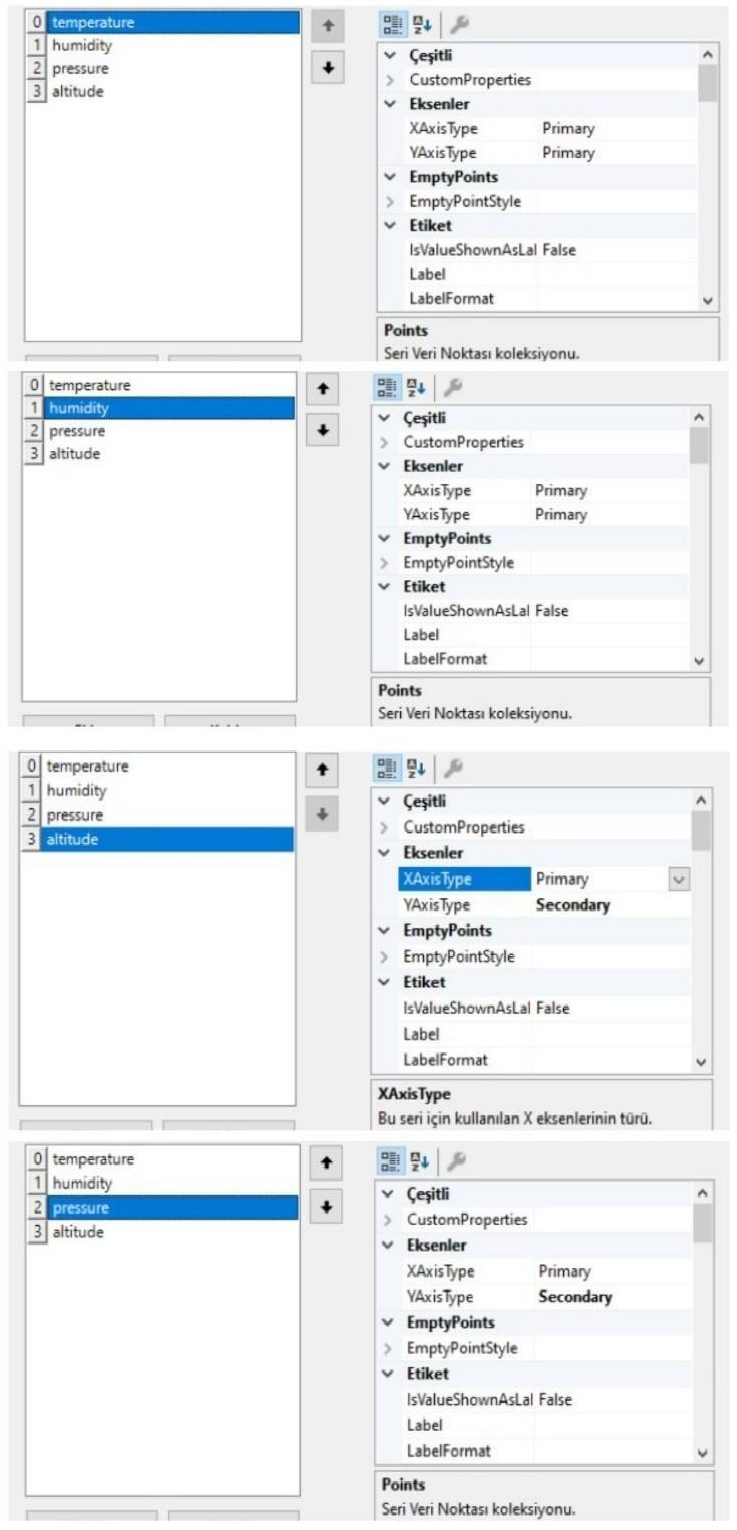


Figure3.2.2Graphic Comment

# SOFTWARE

One of the most important parts of the project is the software part. The proper code must be used to make the project work properly. There is a code for each sensor in our project. We have combined the data from the transmitter to the receiver in a single interface. For this, we used C sharp software and Arduino software. In the source code section, we will talk about the codes in more detail.

# SOURCE CODE

* + 1. CODES OF TRANSMITTER PART

#include <nRF24L01.h> #include <printf.h> #include <RF24.h> #include <RF24\_config.h> #include <DHT.h> #include <Wire.h>

#include <Adafruit\_BMP085.h>

Adafruit\_BMP085 bmp;

#define DHTPIN 5 // pin 0 ve pin 1'e baglamayiniz

#define DHTTYPE DHT11 // DHT11 modulunu tanimliyoruz DHT dht(DHTPIN, DHTTYPE);

RF24 transmit (2,3);

byte address [5] = "00001";

struct package

{

float temperature = 0; float humidity = 0; float pressure = 0; float altitude =0;

};

typedef struct package Package;

Package data;

void setup() { Serial.begin(9600); if(!bmp.begin()){

Serial.println("Could not find a valid BMP085"); while(1){}

}

dht.begin(); transmit.begin();

transmit.openWritingPipe(address); //open writing pipe to address 00001 transmit.setPALevel (RF24\_PA\_MAX); //RF guc outputunu maksimuma ayarliyoruz transmit.setDataRate (RF24\_250KBPS); //datarate'i 250kbps'e ayarliyoruz transmit.setChannel(100); //frekansi 100'e ayarliyoruz

transmit.stopListening();

}

void loop() {

data.temperature = dht.readTemperature(); data.humidity = dht.readHumidity(); data.pressure= bmp.readPressure(); data.altitude=bmp.readAltitude();

transmit.write(&data,sizeof(data)); delay(500);

}

4.1.2.CODES OF RECEIVER PART

#include <nRF24L01.h> #include <printf.h> #include <RF24.h> #include <RF24\_config.h> #include <LiquidCrystal.h> #include <DHT.h> #include <Wire.h>

#include <Adafruit\_BMP085.h>

RF24 receive (2,3);

byte address [5] = "00001"; LiquidCrystal lcd(8,9,4,5,6,7);

#define DHTPIN 5

#define DHTTYPE DHT11 DHT dht(DHTPIN, DHTTYPE);

Adafruit\_BMP085 bmp;

int output\_pin10 = 10; int contrast =90; struct package

{

float temperature = 0; float humidity =0; float altitude = 0; float pressure = 0;

};

typedef struct package Package; Package data;

void setup() { lcd.begin(20,4); lcd.setCursor(0,0); lcd.print("Data Aliniyor");

pinMode(output\_pin10, OUTPUT);

receive.begin(); receive.openReadingPipe(0,address); receive.setPALevel(RF24\_PA\_MIN); receive.setDataRate(RF24\_250KBPS); receive.setChannel(100); receive.startListening();

}

void loop() {

if (receive.available())

{

receive.read(&data, sizeof(data)); lcd.setCursor(0,1); lcd.print(data.temperature); lcd.print("C "); lcd.print(data.humidity); lcd.print(" % "); lcd.setCursor(0,2); lcd.print(data.altitude);

lcd.print(" m"); lcd.setCursor(0,3); lcd.print(data.pressure); lcd.print(" Pa");

}

}

4.1.3)INTERFACE CODE

using System;

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;

using System.IO.Ports; using System.Xml.Schema;

namespace hava

{

public partial class Form1 : Form

{

private string data;

double maxtemperature = 0, maxhumidity = 0, maxpressure = 0, maxaltitude = 0;

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

string[] ports = SerialPort.GetPortNames(); foreach (string port in ports) comboBox1.Items.Add(port); chart.ChartAreas[0].AxisY2.Minimum = 850;

chart.ChartAreas[0].AxisY2.Maximum = 980;

chart.ChartAreas[0].AxisY2.Interval = 15;

chart.ChartAreas[0].AxisY.Minimum = 0;

chart.ChartAreas[0].AxisY.Maximum = 100;

chart.ChartAreas[0].AxisY.Interval = 10;

chart.ChartAreas[0].AxisX.MajorGrid.LineWidth = 0; chart.ChartAreas[0].AxisX.LabelStyle.Format = "d/M/yyyy HH:mm:ss";

}

private void DisplayData\_event(object sender, EventArgs e)

{

DateTime myDateValue = DateTime.Now; label11.Text = myDateValue.ToString(); string[] value = data.Split('/'); textBox1.Text = value[0];

textBox2.Text = value[1]; textBox3.Text = value[2]; textBox7.Text = value[3];

double humidity = Convert.ToDouble(value[1]); double temperature = Convert.ToDouble(value[0]); double pressure = Convert.ToDouble(value[2]); double altitude = Convert.ToDouble(value[3]);

MaksimumBul(temperature, humidity,pressure,altitude);

this.chart.Series[0].Points.AddXY(myDateValue.ToString("d/M/yyyy HH:mm:ss"), temperature); this.chart.Series[1].Points.AddXY(myDateValue.ToString("d/M/yyyy HH:mm:ss"), humidity); this.chart.Series[2].Points.AddXY(myDateValue.ToString("d/M/yyyy HH:mm:ss"), pressure); this.chart.Series[3].Points.AddXY(myDateValue.ToString("d/M/yyyy HH:mm:ss"), altitude);

}

private void MaksimumBul(double temperature, double humidity, double pressure,double altitude)

{

if (temperature > maxtemperature) maxtemperature = temperature;

if (humidity > maxhumidity) maxhumidity = humidity;

if (pressure > maxpressure) maxpressure = pressure;

if (altitude > maxaltitude) maxaltitude = altitude;

textBox4.Text = maxtemperature.ToString(); textBox5.Text = maxhumidity.ToString(); textBox6.Text = maxpressure.ToString(); textBox8.Text = maxaltitude.ToString();

}

private void baglan\_Click\_1(object sender, EventArgs e)

{

try

{

serialPort1.PortName = comboBox1.Text; serialPort1.BaudRate = 9600; serialPort1.Parity = Parity.None; serialPort1.DataBits = 8; serialPort1.StopBits = StopBits.One;

serialPort1.Open();

kes.Enabled = true; baglan.Enabled = false; label9.Text = "Bağlantı sağlandı";

label9.ForeColor = System.Drawing.Color.Green;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Hata");

}

}

private void serialPort1\_DataReceived\_1(object sender, SerialDataReceivedEventArgs e)

{

data = serialPort1.ReadLine();

this.Invoke(new EventHandler(DisplayData\_event));

}

private void Form1\_FormClosed\_1(object sender, FormClosedEventArgs e)

{

if (serialPort1.IsOpen) serialPort1.Close();

}

private void kes\_Click(object sender, EventArgs e)

{

try

{

serialPort1.Close(); kes.Enabled = false; baglan.Enabled = true;

label9.Text = "Bağlantı kesildi"; label9.ForeColor = System.Drawing.Color.Red;

}

catch (Exception ex2)

{

MessageBox.Show(ex2.Message);

}

}

}

}

# CONCLUSION

This report was written about the wireless mini weather station and contains information about how it was formed and what components were used. Wireless mini weather station provides us with data such as instant temperature, humidity, pressure, altitude. First of all, since it consists of two different parts, we placed sensors separately from the receiver and transmitter, checked the correct workings and carried out the data flow. Thanks to the wireless module located in the receiver and transmitter section, we have achieved the data flow. We used Arduinos. Batteries were used as the power source for these C sharp and Arduino languages were used for the programming. As a result, our project has worked without errors and we are receiving healthy data.

* We read our data on the LCD screen and observed the results using ARDUINO codes.
* TEMPERATURE = 25.80 degrees
* HUMIDITY= 41%
* PRESSURE = 90445 Pa
* ALTITUDE= 947 meters (altitude level reference point)

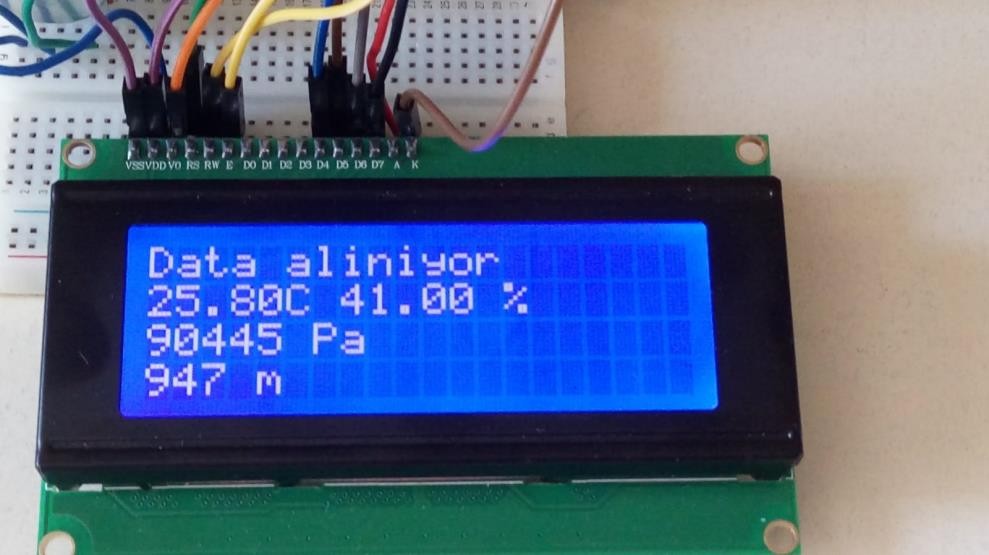


Figure5.1 Results on LCD

# REFERENCES

1. <http://arduinoturkiye.com/dht11-sicaklik-ve-nem-sensorunun-arduino-ile-kullanimi/>
2. [https://support.office.com/tr-tr/article/excel-de-verilerinizi-gantt-grafiğinde-gösterme](https://support.office.com/tr-tr/article/excel-de-verilerinizi-gantt-grafiÄŸinde-gÃ¶sterme)
3. <http://www.sendekodyaz.com/>
4. [www.udemy.com/course/programlamatemelleri](http://www.udemy.com/course/programlamatemelleri)
5. <http://arduinoturkiye.com/arduino-c-net-haberlesmesi/>

# APPENDICES

# COST ANALYSIS

|  |  |  |  |
| --- | --- | --- | --- |
| **Count** | **Item** | **Cost TL / €** | |
| **2** | ARDUNIO-UNO KLON | 70,6 | 9,54 |
| **2** | NRF24L01 | 11,45 | 1,54 |
| **1** | DHT11 | 7,2 | 0,97 |
| **1** | BMP180 | 9 | 1,21 |
| **2** | 9V BATTERY | 22 | 2,97 |
| **1** | DS3231 | 10,41 | 1,62 |
| **2** | BREADBORD | 26 | 3,51 |
| **1** | POT | 5,2 | 0,7 |
| **1** | LCD(4\*20) | 44,49 | 6,01 |
| **1** | JUMBER CABLE SET | 17,12 | 2,29 |
|  | **Total:** | 223,47  TL | 29,79€ |

# GANTT CHART

9 ) R A P O R T P R E P A R İ N G

5.06.2020

8 ) P R E P A R İ N G A P R E S E N T A T İ O N

1.06.2020

7 ) S O F T W A R E W O R K

18.05.2020

14

6 ) I N T E R F A C E D E S İ G N

1.05.2020

17

5 ) M E C H A N İ C A L D E S İ G N

13.04.2020

18

4 ) E Q U İ P M E N T O R D E R

6.04.2020

3 ) A R D U İ N O C O D E W O R K

16.03.2020

21

2 ) E Q U I P M E N T L İ S T

9.03.2020

7

1 ) R E S E A R C H

17.02.2020

20

7

7

4