**A**

**DATA ACQUISITION SYSTEM**

**ON**

**“WEATHER MONITORING SYSTEM”**

FOR PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE

ELECTRONICS SYSTEM DESIGN SUBJECT

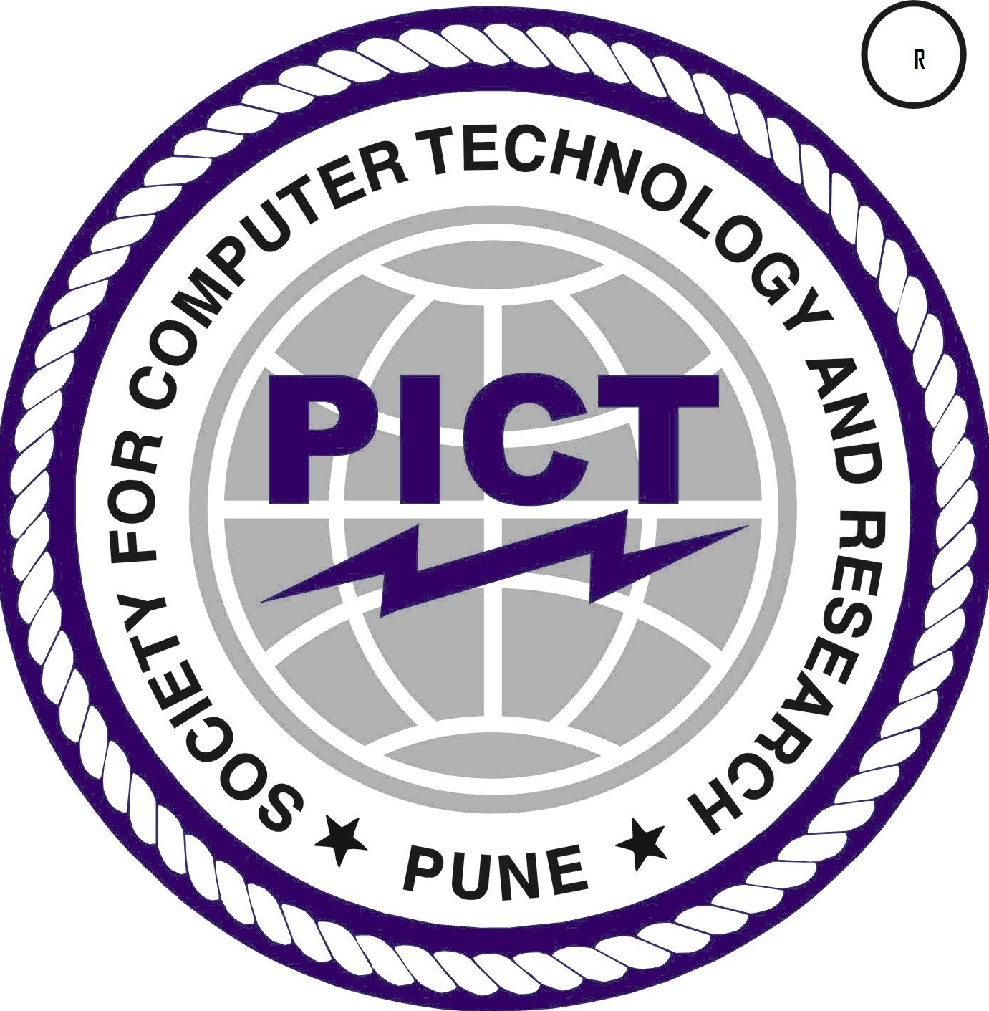
OF T.E. E&TC – 2015 COURSE, SPPU, PUNE

By

**Students of TE-6**

**GUIDE**

**Prof. Sakshi Hosamani**

****

**DEPARTMENT OF**

**ELECTRONICS AND TELECOMMUNICATION ENGINEERING**

**PUNE INSTITUTE OF COMPUTER TECHNOLOGY**

**PUNE – 43**

**ACADEMIC YEAR: 2020 - 2021**

**CHAPTER 1**

* 1. **Background**

How cool would it be to have your own Arduino Weather Station right in your backyard? Sounds fun? Having your own weather station means that you don’t need any more inaccurate results from the weather channel! You can even log the data and play around with it. Set mood lighting according to the weather?

So we’ve got this ultra-low-cost sensor called the DHT11. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air and spits out a digital signal on the data pin. Since the output data isn’t analog, it requires some coding to get the data, but don’t sweat it. It’s got its own library that takes care of the hard parts. It’s fairly simple to use but requires careful timing to grab data. You can only get new data from it once every 2 seconds, but that’s more than enough for our Arduino weather station.

* 1. **Literature Survey**

Previous systems that existed are only on collection of climate data or transmission of these data using ZigBee or GSM or Wi-Fi or some remote mechanism. All these systems, though they measure the same parameters but they lack one common thing and that is accuracy. People need accurate weather conditions of the area they live in. They need to know the weather so that they can thrive and adapt according to it. Other systems collect data and predict tomorrow’s weather data just like that. No pattern, no observations are made. This makes the prediction error prone. This method is applicable only to places where there are not so many weather fluctuations occurring in the area i.e. it is stable throughout. Since normal prediction would fail when the outliers are more. Nowadays, weather stations use heavy instruments to determine the weather of the city. These instruments cost high and their accuracy is not too much to rely on.

* 1. **Proposed System**

To improve the accuracy of the above-mentioned technique, we would be making the weather stations localized. Now we cannot have the whole unit at each and every area. This would incur a lot of expenses and area. To reduce that we would build a mini weather station on top of every building there is in the city. Suppose, if there are 1 million buildings, there will be 1 million mini weather stations incorporated on the top just like a Tata Sky antenna is installed. This would help in collecting data from each area of the city, to be specific each building.

* 1. **Aim of the Project**

The monitoring of weather is really helpful in various applications like in critical scientific systems or for simulation purposes. In other fields like agriculture, disaster management and medical suited environments. Weather sensing is one of the major functions in aerospace applications to check suited weather environments of other planets too also for daily and seasonal reports. The need for this project came from the support of a fact that very low popular devices and instruments are available that can provide you live weather results.

* 1. **Scope and Objectives**

As a future work proposal, it is proposed to apply more sensors such as rain falling meter, pressure sensor, etc. in order to generate a typical robust system. In addition, it has been thought to transfer and demonstrate the realized sensors data wirelessly based on Bluetooth module or Wi-Fi chip over APK smart phone application. Finally, sensor data can be sent to specific phone numbers via GSM module.

* 1. **Technical Approach**

1. Weather forecasting includes various parameters such as Temperature, Humidity, Light, Direction of Wind Speed etc., hence we had decided to build a Weather forecasting Station using Arduino and various sensors.
2. Sensors which were needed to build this project involve ALS-PT19-315C/L177/TR8.
3. We had decided to build the project on the Proteus software in which we involved Arduino Library and built further circuits for further working.

**CHAPTER 2**

**2.1 Introduction**

A weather station is considered a technical method that allows measuring weather parameters based on atmospheric conditions either on the land or on sea for a proposed location with specific devices in order to realize forecasted weather conditions, and to study climate properties.

The collected data allows deciding and confirming the warranty of the proposed chosen location. The weather is appreciated mostly by two parameters, temperature and the humidity. These parameters are fluctuated particularly in the places that have been affected by temperature of the sun radiation and the perpendicular location of the sun that radiates over a specific location based on the latitude of the tropical line.

This project shows a simple way to evaluate the performance of such a significant system. The prosperity and the superiority of Arduino microcontrollers have played a great role in raising this work up. In this project, seven sensors are used to realize weather station attributes to create the proposed database accumulation system. S17021 sensor that measures the Temperature (T) and Humidity (H), MPL3115 sensor that measures Barometric pressure. The Rain Gauge sensor is an additional feature that measures Rainfall in mm.

For Simulation purpose to show the serial communication of the end result that is all the data acquired by the controller from all the connected sensors is displayed on a virtual terminal that gets the data by controlled serially from the serial ports of Arduino. Also, for running the components without getting connected to the computer a power supply should be used which makes this system independent and all the data collected by the Arduino can be transported to some storage device for future study or broadcasting.

* 1. **Block Diagram**

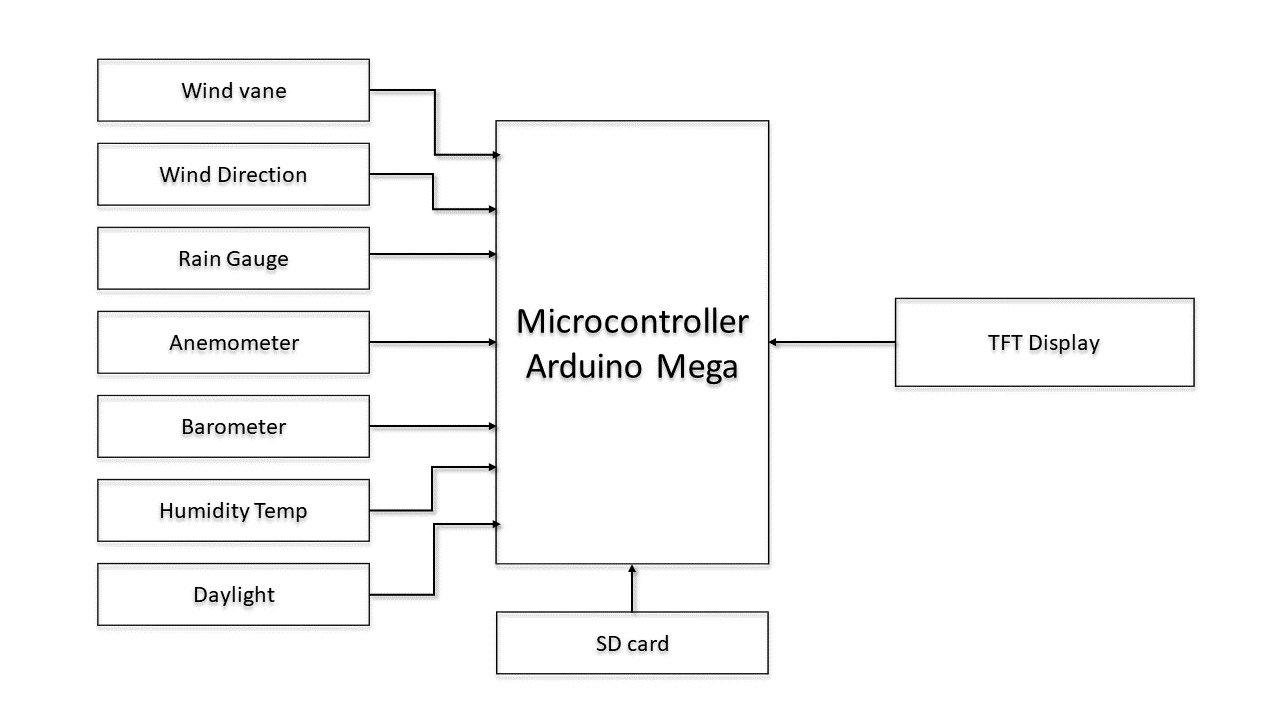


Fig.1. Block schematic of system

* **Wind Vane**

The wind vane is the most complicated of the three sensors. It has eight switches, each connected to a different resistor. The vane’s magnet may close two switches at once, allowing up to 16 different positions to be indicated. An external resistor can be used to form a voltage divider, producing a voltage output that can be measured with an analog to digital converter.



* **Rain gauge**

A rain gauge is a type of instrument used by meteorologists and hydrologists to gather and measure the amount of liquid precipitation over a set period of time. In this project we have used the RG1 sensor.

* **Specifications:**

|  |  |
| --- | --- |
| Resolution In mm | 0.2 |
| Orifice in cm² | 200 |
| Measuring device | Double Tipping buckets |
| Capacity/hour @ Accuracy | up to 50mm @ + 1%  up to 100mm @ + 3% |
| Output signal | one pulse per tip |
| Reed switch | potential free reed switch, fully sealed, made by Meder, Germany; >1 million operations |
| Weight | 0.5 kg / 1.1 lb. |
| Material | anodized aluminum |
| Cable | PUR, shielded, 1m / 3.3ft |
| Connector | 7-pin Binder M9 male |
| Operating temperature | freezing point to +85°C (+180°F) |
| Heating | not available |



* **Anemometer**

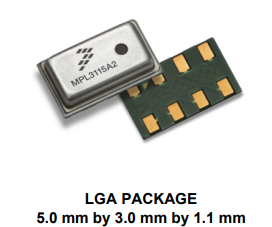
An anemometer is a device used for measuring wind speed and direction. It is also a common weather station instrument. The cup-type anemometer measures wind speed



by closing a contact as a magnet moves past a switch. A wind speed of 1.492 MPH (2.4 km/h) causes the switch to close once per second. The anemometer switch is connected to the inner two conductors of the RJ11 cable shared by the anemometer and wind vane.

* **Barometer:**

The MPL3115A2 employs a MEMS pressure sensor with an I2C interface to provide accurate Pressure/Altitude and Temperature data. The sensor outputs are digitized by a high resolution 24-bit ADC.The device is a low-power, high accuracy digital output altimeter, barometer and thermometer, packaged in a 3 x 5 x 1.1 mm form factor. The complete device includes a sensing element, analog and digital signal processing and an I2C interface.



The internal sensor gives an absolute pressure signal. The absolute pressure signal is processed to provide a scaled pressure or an altitude, depending on the mode selected. The combination of a high performance sensor and the signal processing enable resolution of pressures below 1 Pa and altitude resolution of better than 1 Ft / 0.3m at sea level.



* **Features of MPL3115A2:**

• 1.95 V to 3.6 V Supply Voltage, internally regulated by LDO

• 1.6 V to 3.6 V Digital Interface Supply Voltage

• Fully Compensated internally

• Direct Reading, Compensated – Pressure: 20-bit measurement (Pascals) – Altitude: 20-bit measurement (meters) – Temperature: 12-bit measurement (degrees Celsius)

• Programmable Events

• Autonomous Data Acquisition

• Resolution down to 0.1 m

• 32-Sample FIFO

• Ability to log data up to 12 days using the FIFO

• 1 second to 9 hour data acquisition rate

• I 2C digital output interface (operates up to 400 kHz)

* **Humidity and temperature:**

The Si7021 is a digital relative humidity and temperature sensor that integrates temperature and humidity sensor elements, an analog-to-digital converter, signal processing, calibration, polynomial non-linearity correction, and an I2C interface all in a single chip.



The Si7021 offers a low power, high accuracy, calibrated and stable solution ideal for a wide range of temperature, humidity, and dew-point applications including medical and instrumentation, high reliability automotive and industrial systems, and cost-sensitive consumer electronics.



* **Features:**

Precision Relative Humidity Sensor ± 3% RH (max), 0–80%

RH High Accuracy Temperature Sensor ±0.4 °C (max), –10 to 85 °C

0 to 100% RH operating range

Up to –40 to +125 °C operating range

Wide operating voltage (1.9 to 3.6 V)

Low Power Consumption:150 µA active current

: 60 nA standby current

Factory-calibrated

I 2C Interface

Integrated on-chip heater

3x3 mm DFN Package

Excellent long-term stability

Optional factory-installed cover: Low-profile

: Protection during reflow

: Excludes liquids and particulates

* **Daylight Sensor:**

The ALS-PT19-315C/L177/TR8 is a low cost ambient light sensor, consisting of a phototransistor in miniature SMD. EVERLIGHT ALS series products are a good effective solution to the power saving of display backlighting of mobile appliances, such as the mobile phones, NB and PDAs.



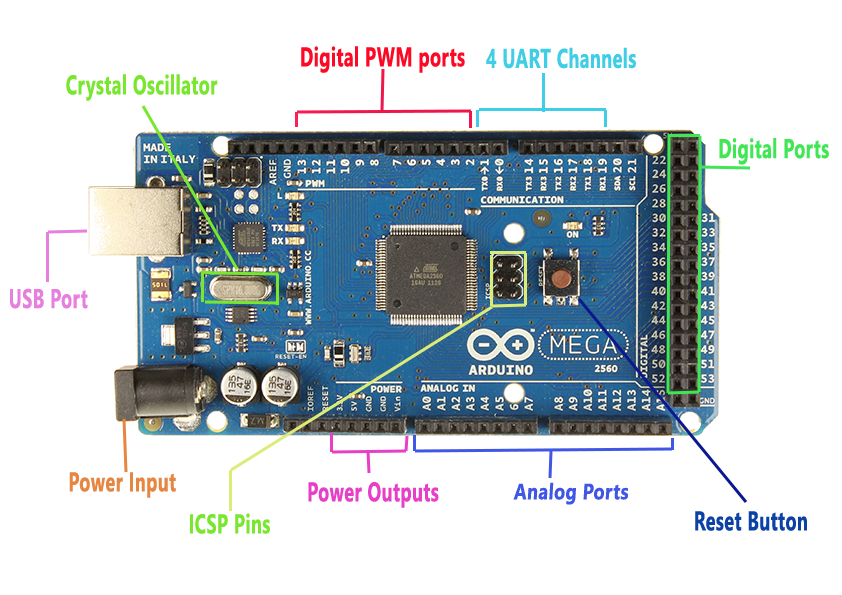
Due to the high rejection ratio of infrared radiation, the spectral response of the ambient light sensor is close to that of human eyes.

* **Features:**
* Close responsively to the human eye spectrum
* Light to Current, analog output
* Good output linearity across wide illumination range
* Low sensitivity variation across various light sources
* Guaranteed temperature performance, -40o C to 85o C
* Wide supply voltage range, 2.5V to 5.5V
* Size: 1.7mm(L)\*0.8mm(W)\*0.6mm(H)
* RoHS compliant and Pb Free package
* **Arduino MEGA 2650:**

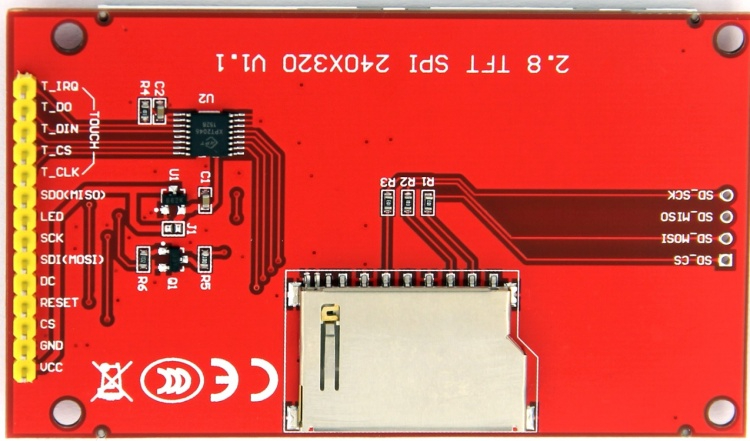
**Arduino** is an open source electronic board that supports both the hardware and software [2]. Arduino is considered the most typical microcontroller that deals with the input modules such as the sensors that transfers the realized data into the Arduino in order to make the suitable decision. Moreover, Arduino microcontroller deals with output modules and devices such as controlling the lights, motors, and the other actuators.

The part responsible for controlling modules is equipped regularly on the board called (ATMEGA) that can be programmed by using the Arduino IDE programming language. Arduino program is written by interfacing the board with computer in order to create programing user interface area to startup controlling tasks properly

The Mega 2560 is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.



So we have displayed all this sensor information in ILI9341 TFT display using Arduino mega

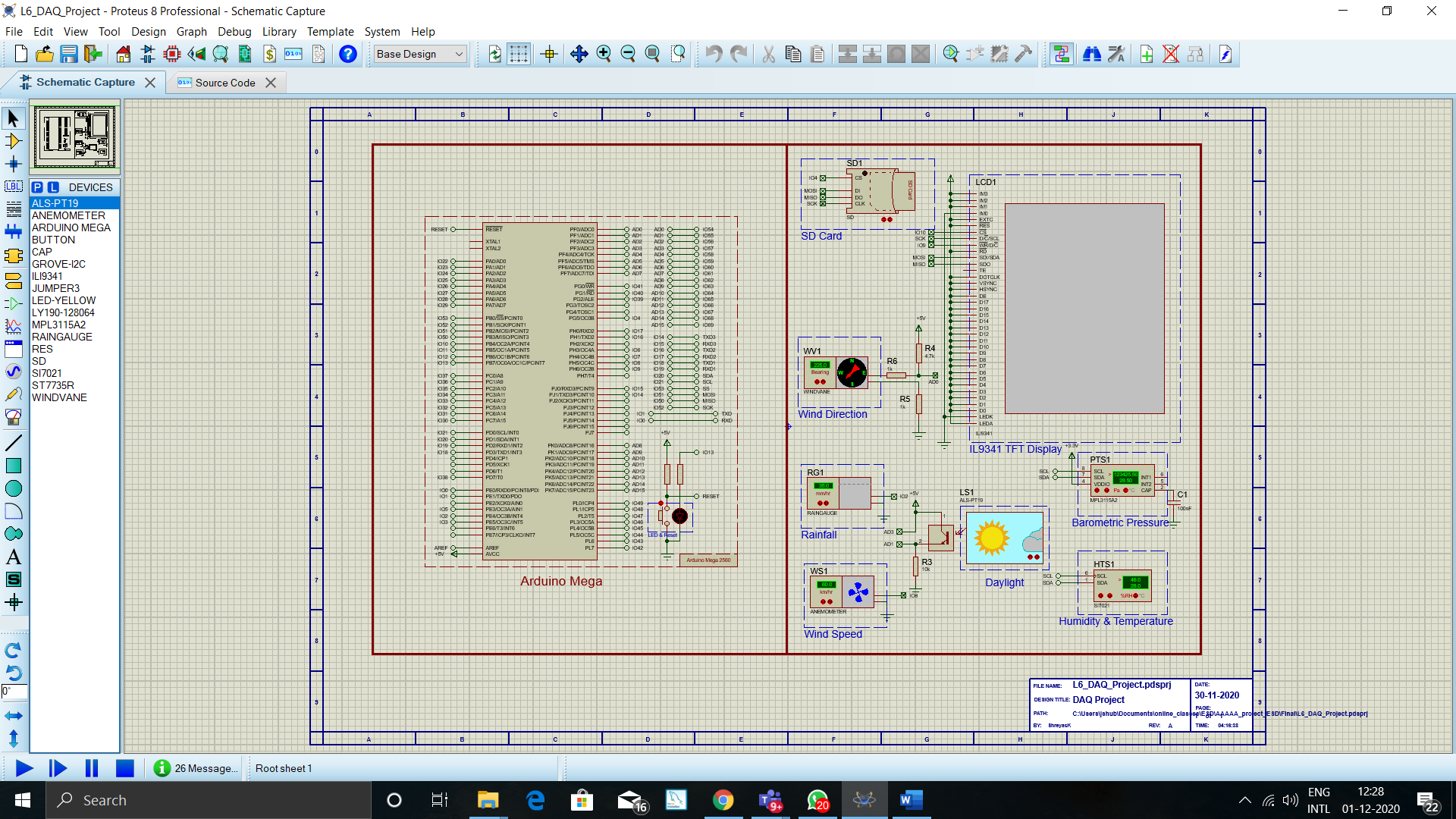
 ****

* **Technical specification:**

|  |  |
| --- | --- |
| Microcontroller | ATmega2560 |
| Operating Voltage | 5V |
| Input Voltage (recommended) | 7-12V |
| Input Voltage (limit) | 6-20V |
| Digital I/O Pins | 54 (of which 15 provide PWM output) |
| Analog Input Pins | 16 |
| DC Current per I/O Pin | 20 mA |
| DC Current for 3.3V Pin | 50 mA |
| Flash Memory | 256 KB of which 8 KB used by bootloader |
| SRAM | 8 KB |
| EEPROM | 4 KB |
| Clock Speed | 16 MHz |
| LED\_BUILTIN | 13 |
| Length | 101.52 mm |
| Width | 53.3 mm |
| Weight | 37 g |

**CHAPTER 3**

**3.1 System Design:**

****

**3.2 Components :**

* Arduino Mega
* SPST push button
* generic non-electrolytic capacitor
* Grove-I2C : Grove Connector
* IL19341 : Graphical TLT LCD with Backlight
* LED
* LY190-128064 : Graphical LCD with SSD1308 Controller
* Sensors
* **List of sensors:**

1) ALS-PT19: Daylight Sensor (For Measuring Cloud Coverage %)

2) SI7021: Temperature & Humidity

3) MPL3115: Barometric Pressure Meter

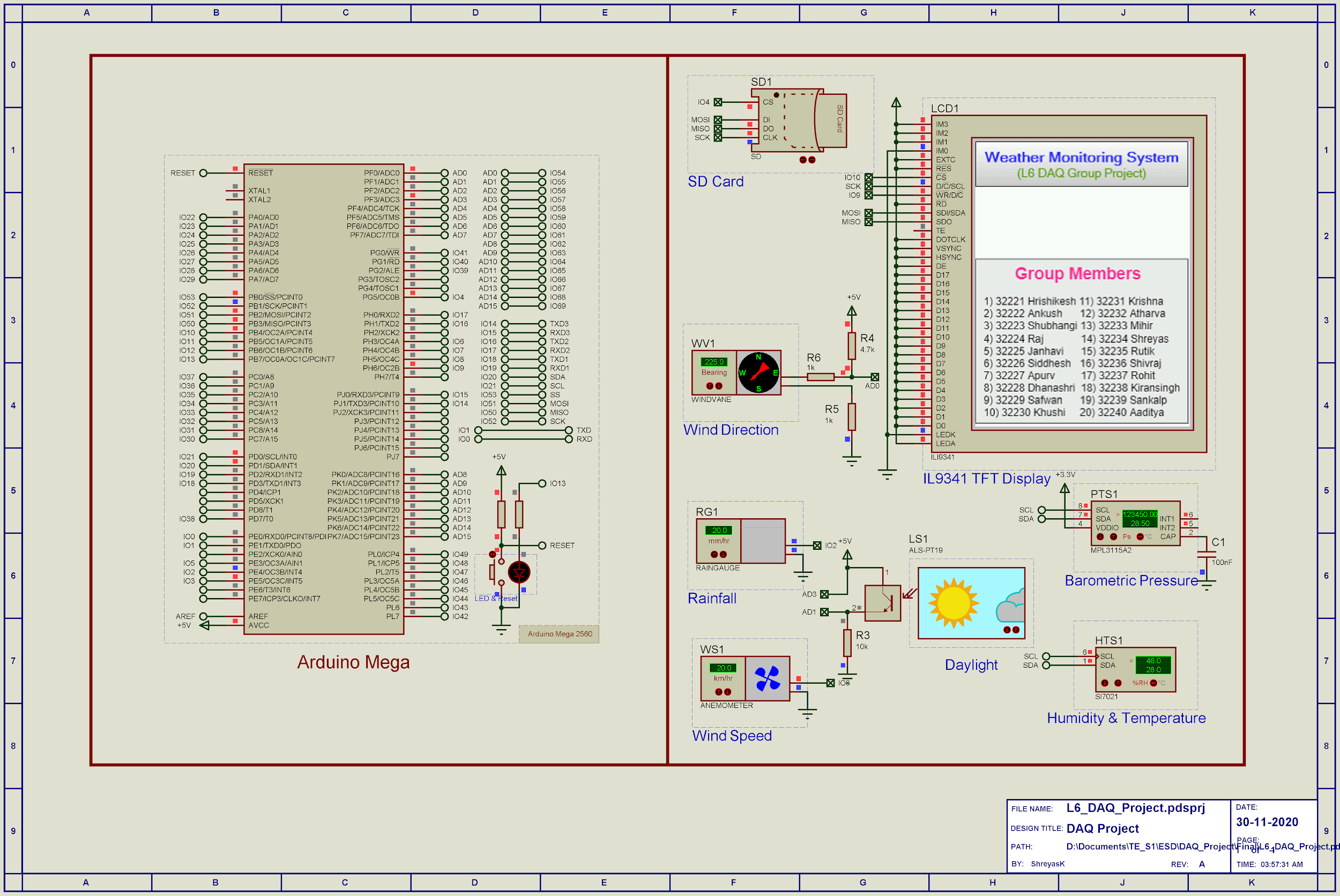
4) Wind Vane (Wind Direction Sensor)

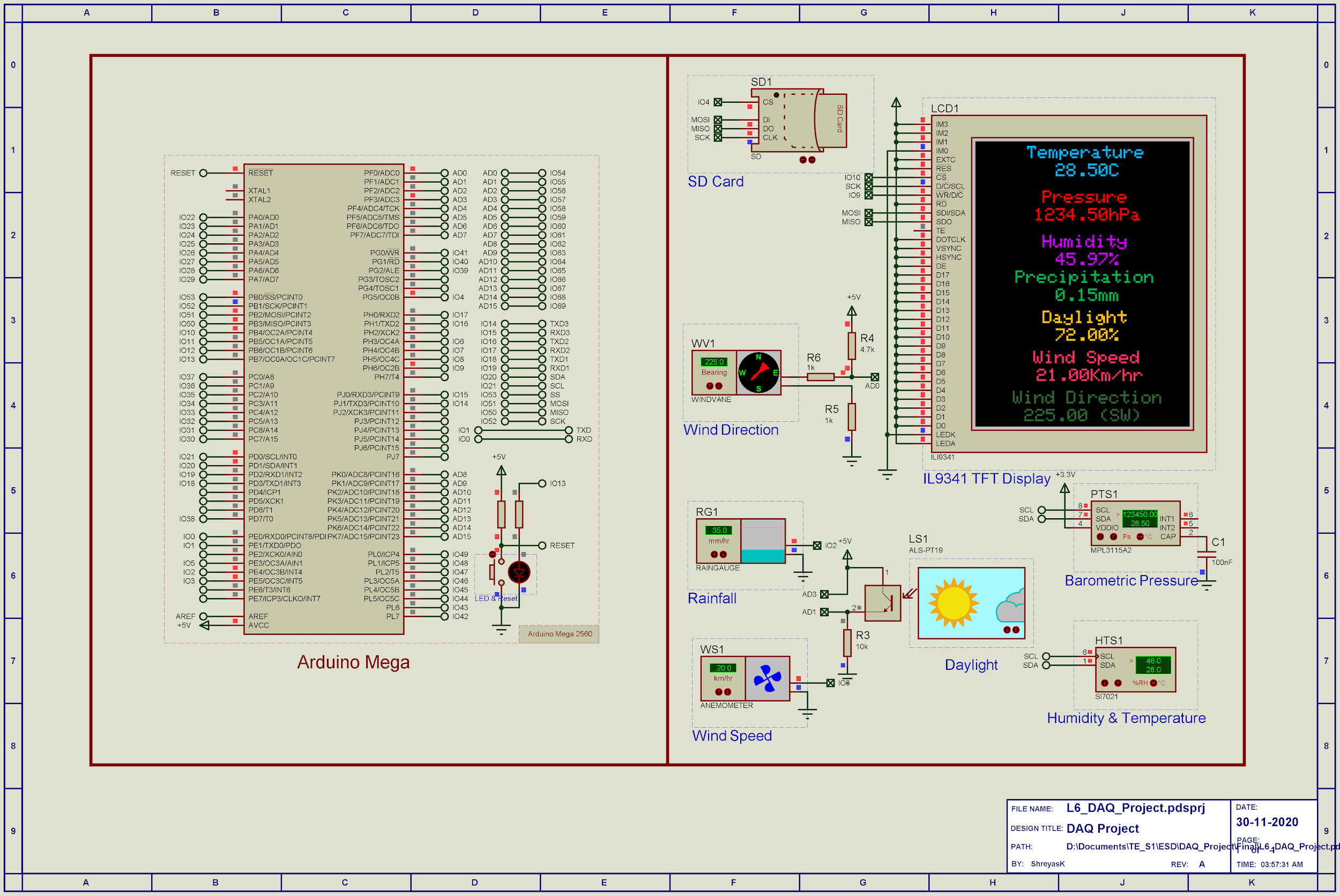
5) Anemometer (Wind Speed Sensor)

6) Rain Gauge: (Measures Rainfall in mm

7) ST7735: TFT Display

**3.3 Snapshots:**

****

****

** **

**3.4 Code:**

/\* Main.ino file generated by New Project wizard

\*

\* Created: Thu Nov 26 2020

\* Processor: Arduino Mega

\* Compiler: Arduino AVR (Proteus)

\*/

// Peripheral Configuration Code (do not edit)

//---CONFIG\_BEGIN---

#pragma GCC push\_options

#pragma GCC optimize ("Os")

#include <core.h> // Required by cpu

#include <cpu.h>

#include <spi.h> // Required by SD1

#include <sd.h> // Required by SD1

#include <filestore.h>

#include <Wire.h> // Required by sensors

#include <Sparkfun\_Weather.h>

#include <wire.h> // Required by display

#include <GFX\_Display.h>

#pragma GCC pop\_options

// Peripheral Constructors

CPU &cpu = Cpu;

WeatherStation sensors = WeatherStation (A0, 3, 2, A3, A1);

GFX\_Display<Adafruit\_ILI9341> display = GFX\_Display<Adafruit\_ILI9341> (10, 9);

FileStore &SD1 = FS;

void peripheral\_setup () {

sensors.begin ();

display.begin ();

SD1.begin (4);

}

void peripheral\_loop() {

sensors.poll ();

}

//---CONFIG\_END---

float Humidity, Temperature, WindSpeed = -1, Daylight, Pressure, Precipitation, WindDirection, DewPoint;

volatile int pulseCount = 0;

volatile int mmCount = 0;

void setup()

{

peripheral\_setup();

display.fillScreen(0xFFFF);

detachInterrupt(digitalPinToInterrupt(2));

attachInterrupt(digitalPinToInterrupt(2), mmCounterISR, FALLING);

display.drawBitmap("splash1.bmp",0,0);

display.drawBitmap("splash2.bmp",0,130);

delay(5000);

display.setTextSize(2);

sensors.reset();

}

void loop()

{

peripheral\_loop();

Temperature = sensors.getTemperature();

Humidity = sensors.getHumidity();

Pressure = sensors.getPressure();

Daylight = getDayLightPercent();

Precipitation = mmCount\*0.05;

WindDirection = getWindDirectionAngle();

display.fillScreen(0);

display.setCursor(58, 5);

display.setTextColor(0x059E);

display.print().arg("Temperature").end();

display.setCursor(90, 25);

display.setTextColor(0x059E);

display.print().arg(Temperature).end();

display.print().arg("C").end();

display.setCursor(75, 55);

display.setTextColor(0xF800);

display.print().arg("Pressure").end();

display.setCursor(66, 75);

display.setTextColor(0xF800);

display.print().arg(Pressure).end();

display.print().arg("hPa").end();

display.setCursor(75, 105);

display.setTextColor(0xC81F);

display.print().arg("Humidity").end();

display.setCursor(90, 125);

display.setTextColor(0xC81F);

display.print().arg(Humidity).end();

display.print().arg("%").end();

display.setCursor(45, 145);

display.setTextColor(0x058A);

display.print().arg("Precipitation").end();

display.setCursor(90, 165);

display.setTextColor(0x058A);

display.print().arg(Precipitation).end();

display.print().arg("mm").end();

display.setCursor(75, 190);

display.setTextColor(0xFE00);

display.print().arg("Daylight").end();

display.setCursor(90, 210);

display.setTextColor(0xFE00);

display.print().arg(Daylight).end();

display.print().arg("%").end();

display.setCursor(65, 235);

display.setTextColor(0xF98C);

display.print().arg("Wind Speed").end();

display.setCursor(69, 255);

display.setTextColor(0xF98C);

if (WindSpeed == -1)

{

display.print().arg("Measuring...").end();

}

else

{

display.print().arg(WindSpeed).end();

display.print().arg("Km/hr").end();

}

display.setCursor(42, 280);

display.setTextColor(0x3326);

display.print().arg("Wind Direction").end();

display.setCursor(55, 300);

display.setTextColor(0x3326);

display.print().arg(WindDirection).end();

display.print().arg(" (" + getWindDirectionStr(WindDirection) + ")").end();

WindSpeed = map(pulseCount, 0, 312, 0, 150);

pulseCount = 0;

detachInterrupt(digitalPinToInterrupt(3));

attachInterrupt(digitalPinToInterrupt(3), pulseCounterISR, RISING);

delay(5000);

detachInterrupt(digitalPinToInterrupt(3));

}

String getWindDirectionStr(int angle)

{

if ((angle >= 0 && angle < 23) || (angle >= 338 && angle < 360))

return "N";

if (angle >= 23 && angle < 68)

return "NE";

if (angle >= 68 && angle < 113)

return "E";

if (angle >= 113 && angle < 158)

return "SE";

if (angle >= 158 && angle < 203)

return "S";

if (angle >= 203 && angle < 248)

return "SW";

if (angle >= 248 && angle < 293)

return "W";

if (angle >= 293 && angle < 338)

return "NW";

return "??";

}

int getWindDirectionAngle()

{

int adc = analogRead(A0);

if (adc < 380)

return (113);

if (adc < 393)

return (68);

if (adc < 414)

return (90);

if (adc < 456)

return (158);

if (adc < 508)

return (135);

if (adc < 551)

return (203);

if (adc < 615)

return (180);

if (adc < 680)

return (23);

if (adc < 746)

return (45);

if (adc < 801)

return (248);

if (adc < 833)

return (225);

if (adc < 878)

return (338);

if (adc < 913)

return (0);

if (adc < 940)

return (293);

if (adc < 967)

return (315);

if (adc < 990)

return (270);

return -1;

}

int getDayLightPercent()

{

return map(analogRead(A1), 0, 422, 0, 100);

}

void pulseCounterISR()

{

pulseCount++;

}

void mmCounterISR()

{

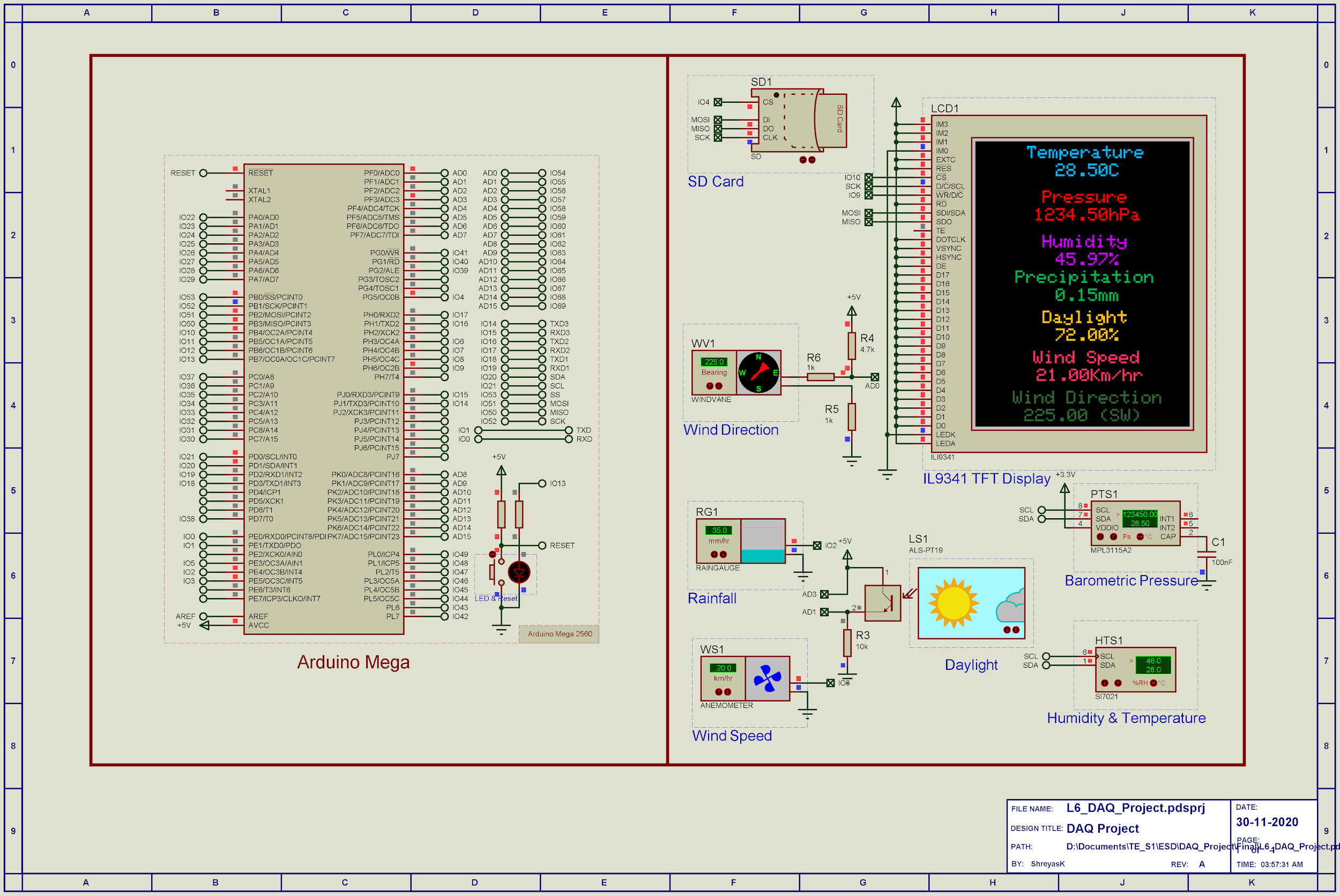
mmCount++;

**CHAPTER 4**

**Observation and Result**

The proposed system attributes are consisted of four different elements identified by Temperature, Humidity , WindSpeed , Barometric pressure, Rainfall, Daylight and Wind direction as identified before. The data given in these Six attributes are analyzed as follows:

**4.1 Result:**



****

* **Real time Arduino Weather Station**



**4.2 Conclusion**

We built a weather station that measures temperature, relative humidity and air pressure. The station was programmed to read values from the sensor when a user asks for them.

1. We can determine if the weather is HOT, NORMAL, or COLD based on the air , temperature and humidity.
2. This weather monitoring system will provide farmers, pharmacists, event planners and others with accurate information to guide them to take appropriate action.
3. The fundamentals of Data acquisition system was understood
4. Simple DAQ system was designed using microcontroller
5. The DAQ system was implemented and simulated in Proteus

**4.3 Future Scope**

As a future work proposal, it is proposed to apply more sensors such as rain falling meter , pressure sensor, etc. in order to generate a typical robust system. In addition, it has been thought to transfer and demonstrate the realized sensors data wirelessly based on Bluetooth module or Wi-Fi chip over APK smart phone application. Finally, sensor data can be sent to specific phone numbers via GSM module.