PROJECT REPORT

On

LAN/Cloud Based System Control and Data Monitoring with ARDUINO UNO

Submitted By:

KUNAL KUMAR (NIT Patna)

B.TECH (E.C.E-6th Semester)

(Training Period: 28 MAY 2018 to 5 JULY 2018)

Guided By:

Surendra Mahavir Jain

Scientist 'E'

ACKNOWLEDGMENT

I thankfully acknowledge and pay my regards to Shri A.K. Saxena, Director of ADRDE, Agra for giving me an opportunity to work in a highly conducive environment of the Aerial Delivery Research and Development Establishment, Ministry of Defence, Govt. of India, Agra.

Shri. Surendra Mahavir Jain Sir (Scientist 'E'), provided me guidance continuously during the project work at ADRDE, without which the work could not have been finished successfully and I am grateful to him.

It goes without saying that he was a constant source of inspiration and guided me to successful completion of project. He supported and provided me with valuable information and components regarding the project and without him this in-depth analysis could not have been possible.

I express my sincere gratitude to all those within and outside ADRDE related with my work directly or indirectly during the course of completion of this project.

CERTIFICATE

This is to certify that Kunal Kumar, a student of National Institute of Technology Patna has successfully completed training of six weeks at Aerial Delivery Research Development and Establishment (ADRDE), Agra on the "LAN/Cloud Based System Control and Data Monitoring with ARDUINO UNO".

S. M. Jain
(Scientist 'E')
(ADRDE, Agra)

Contents

1. INTRODUCTION

1.1 ADRDE:

Aerial Delivery Research & Development Establishment (ADRDE), a pioneer establishment of ministry of Defence, Government of India engaged in design & development of Flexible Aerodynamics Decelerators commonly known as '**parachute**' and the allied technologies.

1.2 Project Overview:

This project is based on the concept of Internet of Things (IOT), which is connecting the Electronic devices to internet or Local Area Network (LAN). The purpose of this project was to make an end to end user friendly interface(UI) for controlling different system remotely using LAN and displaying the sensor data through cloud platform (THINGSPEAK) powered by Mathworks cloud platform. The Micro-controller used in this project is ARDUINO UNO.
The programming of board is written in Arduino programming language which resembles with 'C' programming language and the web scripts are written in HTML, CSS, and JAVASCRIPT.

Applications of this project:

a. It can be used to control a system (Switch ON and OFF) remotely by just accessing the IP Address of the controller with the help of webpage (UI) or with the webpage if hosted.

b. We can monitor the data coming of the sensor attached to the Arduino Board using cloud platform and continuously plot the different reading coming from the board to cloud from anywhere in the world by just accessing the account with which the channel is connected at the cloud.

2. INTERNET OF THINGS

2.1 Introduction:

Wikipedia Says

The Internet of Things (IoT) is the network of physical objects—devices, vehicles, buildings and other items which are embedded with electronics, software, sensors, and network connectivity, which enables these objects to collect and exchange data.

History of IOT

Kevin Ashton (born 1968) is a British technology pioneer He cofounded the Auto-ID Centre at the Massachusetts Institute of Technology (MIT), which created a global standard system for RFID and other sensors. He is known for inventing the term "The Internet of Things" to describe a system where the Internet is connected to the physical world via ubiquitous sensors.

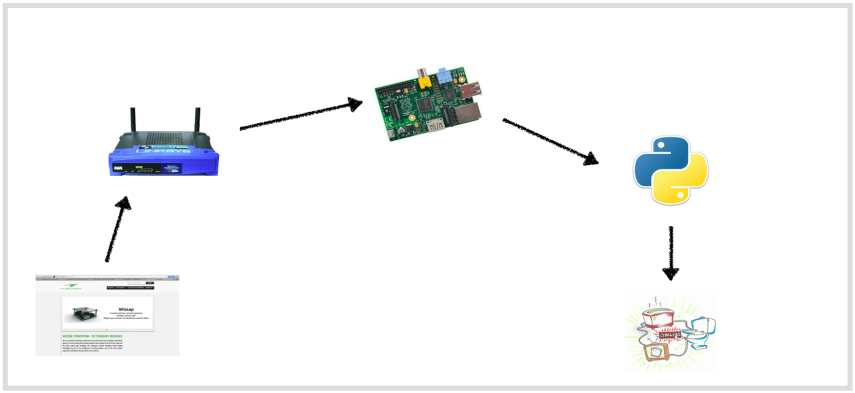
Prospects of IoT as stated by Leading Tech Players:

"It will be bigger than anything that's ever been done in high tech, It will change the way people live, work and play" -John Chambers, CEO of Cisco Systems Inc.

What's in it for future engineers?

Cisco has estimated that 50 billion devices will be connected to the Internet by next year, and 50 billion by 2020. IoT is expected to be a $14 trillion industry by 2020. One of the biggest challenges for expatiation of IoT is on human resource. More and more companies will require skilled engineers who are trained in the field of loT. This makes it a great opportunity for engineers who have knowledge of this domain.

2.2 Data Flow Diagram for IOT Devices:



EXPLANATION:

- **User Interface application**-Is a user oriented application, with an GUI (Graphical user interface), that allows users to control or monitor IoT enabled devices remotely over a local area network (LAN) or internet. E.g.: Mobile apps, web apps etc.

- **Network components**-Includes all networking components, like routers, switches, hubs, etc. which help in transporting information to and from the User Interface Application to the IoT enabled-devices.

- **Server**- The computing machine which hosts the User Interface Application. In some cases it may be the machines that controls or monitors “the things” directly or is “the thing” itself.

- **Hardware control script**-The programming language used to communicate between the server and the IoT enabled device. Any programming language capable of communicating directly to controller hardware can be used.

2.3 THE THING:

The "Thing" is an embedded computing device (or embedded system) that transmits and receives information over a network (need not be able to interface with internet directly) for the purpose of controlling another device or interacting with a user. A Thing is also a microcontroller or microprocessor-based device.

Hence a simple chair, TV, fan, microwave, fridge, sprinkler, bulb etc., (the list goes on) on their own cannot be called "Things" with respect to IoT because:

- Most of day to day things do not have any embedded systems (processing capability) Example: bed, chair, fan, and bulb.

- Even if they do have embedded systems built in, they do not have the capabilities to transmit and receive information over a network. E.g. washing machine, microwave, electric stoves.

The "Thing" hence should provide one or all of the below services:

1. Identification and info storage (RFID tags, MAC address)

2. Information collection (Sensor networks, store sensor values)

3. Information processing (Understanding commands, filtering data)

4. Communications (Transmit and receive messages)

5. Actuation (Switch control, motor control)

Hence for anything to be called a "Thing" in IoT we need to add one or all of the above features to day to day things externally or have them inbuilt. That means you have to add an embedded device to say a chair, window, ceiling fan or bulb for them to have the above mentioned capabilities. Some systems which already have the above features are smart TV, all smart phones, smart thermostats, desk computers/laptops, etc.

What is the importance of these "Things"?

The "Thing" plays a major role in this concept. So much so that the world is looking to widely adopt IPv6 protocol especially to address the things. It is estimated that by

2020, 20 Billion things will come online i.e. be interconnected via the World Wide Web. But the fact still remains that the ideology "one size fits all" cannot be implemented for the things, the simple reason being is each "Thing" will have to be designed as per requirements of the specific application and such an approach will not be cost or performance optimized enough to satisfy the needs of this market. Hence, to minimize costs and optimize throughput the "Thing" becomes of utmost importance because of their sheer staggering numbers.

How do we make these "Things"?

One understanding we have to come to before we go forward is that the "Thing" has to have either a microcontroller or microprocessor. So first we have to choose the Microcontroller. Factors to consider in selection would be:

- Memory size
- Number of General Purpose input output pins
- Peripheral-communication (SPI,I2C,USART)
- Communication-capabilities (Wired E.g. : Ethernet, Wireless E.g. : Wi-Fi, Bluetooth, Zigbee, IR etc.)

All above components can be present on a single chip or can be externally interfaced to a microprocessor.

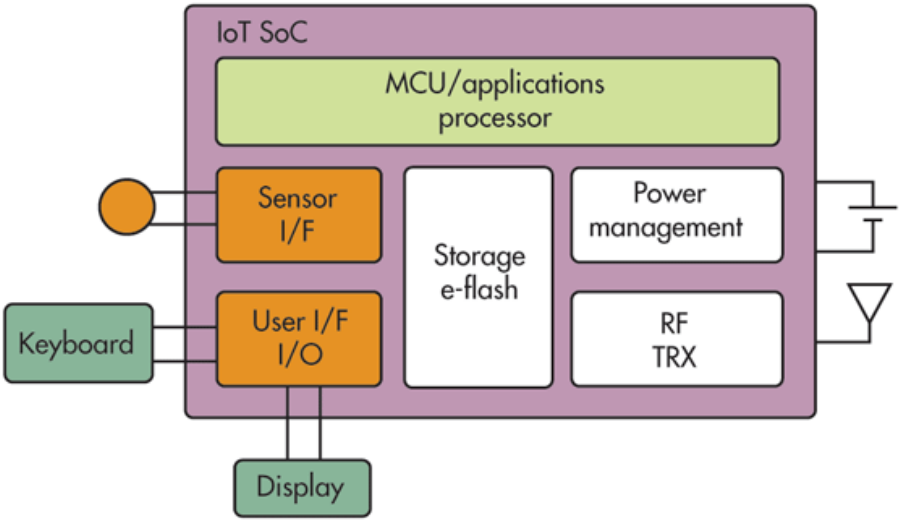


Image Source: datasciencebe.com

Once the microcontroller is selected, the next thing to look at is the type of **SENSOR** you want to use.

Again selection of sensor is purely application specific. All we need to research here is as to what real life quantity we need to measure e.g.: light, sound, pressure, gas, etc. and find a device (transducer) that can convert it into an electrical signal (current or voltage) which can be either digital or analog. If the signal is analog then your embedded system will always need an Analog to Digital converter.

There are already a myriad of sensors available today with a bit of research needs to be done to successfully interface them with your microcontroller.

Another important thing to consider is sensor sensitivity and power consumed.

In case we want the "Thing" to control its physical environment, we need some form of **ACTUATORS**. Again while selecting each is application specific. These include motors, relays, switches, valves etc.

Finally after we have finalized and settled with the above components we need to now look at how these "Things" can **COMMUNICATE**. Now they can have on board wired or wireless communication capabilities or can have added modules which do the communication process on their command (E.g.: Bluetooth or Wi-Fi dongles).

3. HARDWARE DESCRIPTION

3.1 Introduction:

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino Boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the arduino programming language (based on Wiring), and the Arduino software (IDE), based on processing.

Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike.

Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IOT (Internet Of Things) applications, wearable, 3D printing, and embedded environments. All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt them to their particular needs. The software too is open-source, and it is growing through the contributions of users worldwide.

Why Arduino?

Thanks to its simple and accessible user experience, Arduino has been used in thousands of different projects and applications. The Arduino software is easy-to-use for beginners, yet flexible enough for advanced users. It runs on Mac, Windows, and Linux. Teachers and students use it to build low cost scientific instruments, to prove chemistry and physics principles, or to get started with programming and robotics. Designers and architects build interactive prototypes, musicians and artists use it for installations and to experiment with new musical instruments. Makers, of course, use it to build many of the projects exhibited at the Maker Faire, for example. Arduino is a key tool to learn new things. Anyone - children, hobbyists, artists, programmers - can start tinkering just following the step by step instructions of a kit, or sharing ideas online with other members of the Arduino community.

There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, and many others offer similar functionality. All of these tools take the messy details of microcontroller programming and wrap it up in an easy-to-use package. Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

- Inexpensive - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $50
- Cross-platform - The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.
- Simple, clear programming environment - The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with how the Arduino IDE works.
- Open source and extensible software - The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. Similarly, you can add AVR-C code directly into your Arduino programs if you want to.
- Open source and extensible hardware - The plans of the Arduino boards are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

3.2 ARDUINO UNO BOARD OVERVIEW:

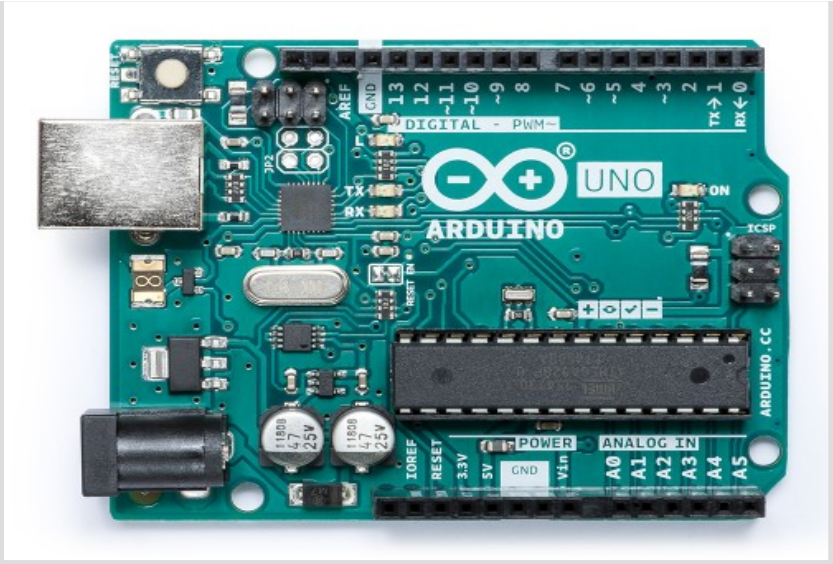
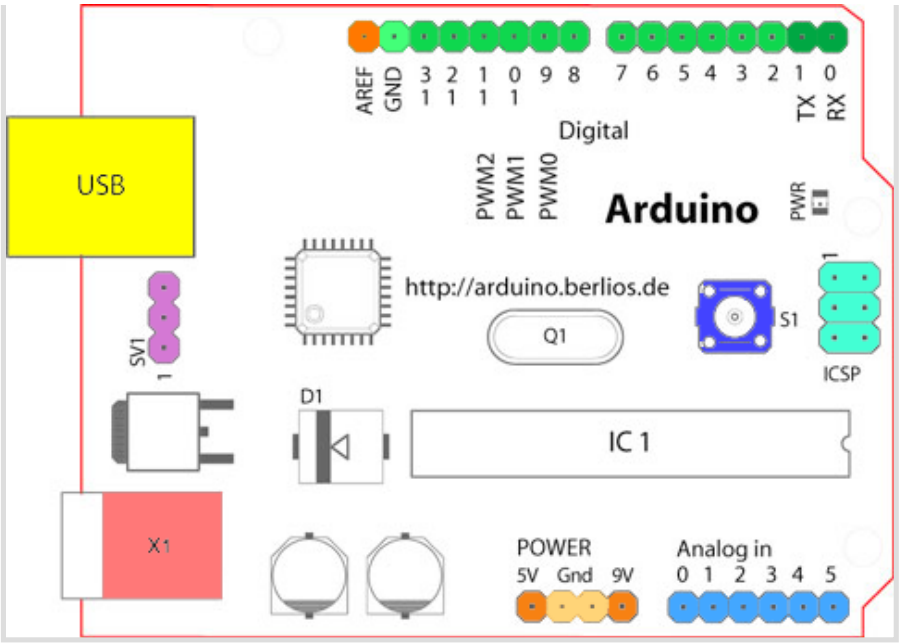


fig. : Arduino Uno Board

The UNO is the best board to get started with electronics and coding. **Arduino Uno** is a microcontroller board based on the ATmega328P . It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, 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. You can tinker with your UNO without worrying too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again. "Uno" means one in Italian and was chosen to mark the release of Arduino Software (IDE) 1.0. The Uno board and version 1.0 of Arduino Software (IDE) were the reference versions of Arduino, now evolved to newer releases. The Uno board is the first in a series of USB Arduino boards, and the reference model for the Arduino platform; for an extensive list of current, past or outdated boards.

Pin Description:

Pin Category	Pin Name	Details
Power	Vin, 3.3V, 5V, GND	Vin: Input voltage to Arduino when using an external power source. 5V: Regulated power supply used to power microcontroller and other components on the board. 3.3V: 3.3V supply generated by on-board voltage regulator. Maximum current draw is 50mA. GND: ground pins.
Reset	Reset	Resets the microcontroller.
Analog Pins	A0 – A5	Used to provide analog input in the range of 0-5V
Input/output Pins	Digital Pins 0 - 13	Can be used as input or output pins.
Serial	0(Rx), 1(Tx)	Used to receive and transmit TTL serial data.
External Interrupts	2, 3	To trigger an interrupt.
PWM	3, 5, 6, 9, 11	3, 5, 6, 9, 11
SPI	10 (SS), 11 (MOSI), 12 (MISO) and 13 (SCK)	Used for SPI communication
Inbuilt LED	13	To turn on the inbuilt LED.
TWI	A4 (SDA), A5 (SCA)	Used for TWI communication.



Microcontrollers

ATmega328P (used on most recent boards) Digital I/O Pins 14 (of which 6 provide PWM output) Analog Input Pins 6 (DIP) or 8 (SMD) DC Current per I/O Pin 40 mA Flash Memory 32 KB SRAM 2 KB EEPROM 1KB	*ATmega168* (used on most Arduino Diecimila and early Duemilanove) Digital I/O Pins 14 (of which 6 provide PWM output) Analog Input Pins 6 (DIP) or 8 (SMD) DC Current per I/O Pin 40 mA Flash Memory 16 KB SRAM 1 KB EEPROM 512 bytes	*ATmega8* (used on some older board) Digital I/O Pins 14 (of which 3 provide PWM output) Analog Input Pins 6 DC Current per I/O Pin 40 mA Flash Memory 8 KB SRAM 1 KB EEPROM 512 bytes

Digital Pins

In addition to the specific functions listed below, the digital pins on an Arduino board can be used for general purpose input and output via the pinMode(), digitalRead(), and digitalWrite() commands. Each pin has an internal pull-up resistor which can be turned on and off using digitalWrite() (w/ a value of HIGH or LOW, respectively) when the pin is configured as an input. The maximum current per pin is 40 mA.

- Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. On the Arduino Diecimila, these pins are connected to the corresponding pins of the FTDI USB-to-TTL Serial chip. On the Arduino BT, they are connected to the corresponding pins of the WT11 Bluetooth module. On the Arduino Mini and LilyPad Arduino, they are intended for use with an external TTL serial module (e.g. the Mini-USB Adapter).
- External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.
- PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite() function. On boards with an ATmega8, PWM output is available only on pins 9, 10, and 11.
- BT Reset: 7. (Arduino BT-only) Connected to the reset line of the Bluetooth module.
- SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language.
- LED: 13. On the Diecimila and LilyPad, there is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

Analog Pins

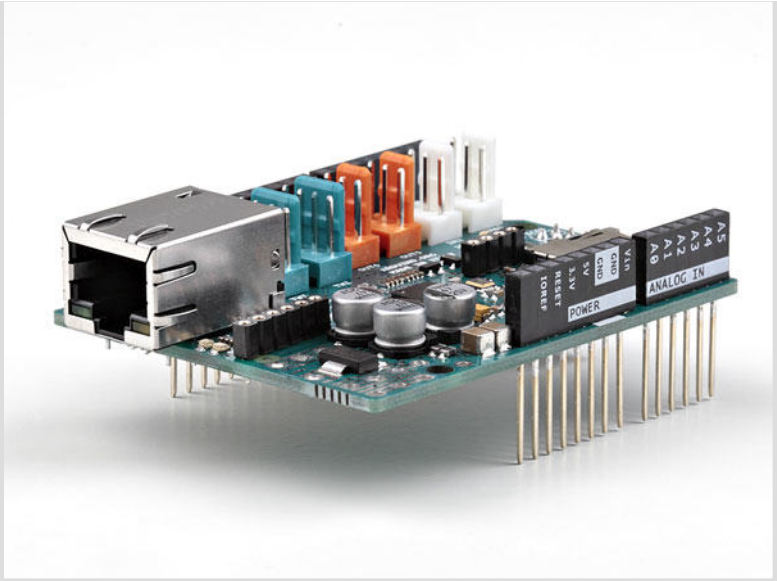
In addition to the specific functions listed below, the analog input pins support 10-bit analog-to-digital conversion (ADC) using the analogRead() function. Most of the analog inputs can also be used as digital pins: analog input 0 as digital pin 14 through analog input 5 as digital pin 19. Analog inputs 6 and 7 (present on the Mini and BT) cannot be used as digital pins.

- I^2C: 4 (SDA) and 5 (SCL). Support I^2C (TWI) communication using the Wire library .

Power Pins

- VIN (sometimes labelled "9V"). The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin. Note that different boards accept different input voltages rang. Also note that the LilyPad has no VIN pin and accepts only a regulated input.
- 5V. the regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- 3V3. (Diecimila-only) A 3.3 volt supply generated by the on-board FTDI chip.
- GND. Ground pins.

3.3 ARDUINO ETHERNET SHIELD:



The Arduino Ethernet shield allows an Arduino board to connect to the internet using the Ethernet library and to read and write an SD card using the SD library. This shield is fully compatible with the former version, but relies on the newer W5500 chip. Depending on the shield version you have, you need to use the proper library, as documented in the Ethernet library page. To use the shield, mount it on top of an Arduino board (e.g. the Uno). To upload sketches to the board, connect it to your computer with a USB cable as you normally would. Once the sketch has been uploaded, you can disconnect the board from your computer and power it with an external power supply. Connect the shield to your computer or a network hub or router using a standard Ethernet cable (CAT5 or CAT6 with RJ45 connectors). Connecting to a computer may require the use of a cross-over cable (although many computers, including all recent Macs can do the cross-over internally).

Network Settings

The shield must be assigned a MAC address and a fixed IP address using the Ethernet.begin() function. A MAC address is a globally unique identifier for a particular device. Current Ethernet shields come with a sticker indicating the MAC address you should use with them. For older shields without a dedicated MAC address, inventing a random one should work, but don't use the same one for multiple boards. Valid IP addresses depend on the configuration of your network. It is possible to use DHCP to dynamically assign an IP to the shield. Optionally, you can also specify a network gateway and subnet.

SD Card

The latest revision of the Ethernet Shield includes a micro-SD card slot, which can be interfaced with using the SD library.

3.4 LINEAR DISPLACEMENT SENSOR:

Fig: Liner Displacement sensor

The SLS320 displacement sensors range is designed to provide maximum performance benefits within a body diameter of 32mm, with stroke lengths from 250 to 1600mm. With a choice of mounting options and accessories, this sensor is ideally suited to a wide range of heavier duty industrial applications, for medium to long stroke linear position sensing.
Using the proven benefits of Hybrid Track Technology and including a number of unique design features, the SLS320 is ideally suited to high volume OEM manufacturers, where high performance and reliability with competitive pricing and rapid dispatch has paramount importance.

Options:
Compact shaft: Compact shaft will reduce dimension D by 50mm
Integral shaft seal - IP 66: Designed to accept integral shaft seal to give IP66 rating
Cabled socket: 1m or 10m cabled socket assemblies available
Mounting: Body clamp or flange mounting kits can be supplied
Protective sleeve: For all stroke lengths - self aligning bearings only. See ordering code

3.5 SERVO MOTOR:

A **servomotor** is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity and acceleration.[1] It consists of a suitable motor coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors.

- Servomotors are not a specific class of motor although the term servomotor is often used to refer to a motor suitable for use in a closed-loop control system.
- Servomotors are used in applications such as robotics, CNC machinery or automated manufacturing.

MECHANISM:

A servomotor is a closed-loop servomechanism that uses position feedback to control its motion and final position. The input to its control is a signal (either analog or digital) representing the position commanded for the output shaft.

The motor is paired with some type of encoder to provide position and speed feedback. In the simplest case, only the position is measured. The measured position of the output is compared to the command position, the external input to the controller. If the output position differs from that required, an error signal is generated which then causes the motor to rotate in either direction, as needed to bring the output shaft to the appropriate position. As the positions approach, the error signal reduces to zero and the motor stops.

The very simplest servomotors use position-only sensing via a potentiometer and bang-bang control of their motor; the motor always rotates at full speed (or is stopped). This type of servomotor is not widely used in industrial motion control, but it forms the basis of the simple and cheap servos are used for radio-controlled models.

More sophisticated servomotors use optical rotary encoders to measure the speed of the output shaft and a variable-speed drive to control the motor speed. Both of these enhancements, usually in combination with a PID control algorithm, allow the servomotor to be brought to its commanded position more quickly and more precisely, with less overshooting.

Servomotors are generally used as a high-performance alternative to the stepper motor. Stepper motors have some inherent ability to control position, as they have built-in output steps. This often allows them to be used as an open-loop position control, without any feedback encoder, as their drive signal specifies the number of steps of movement to rotate, but for this the controller needs to 'know' the position of the stepper motor on power up. Therefore, on first power up, the controller will have to activate the stepper motor and turn it to a known position, e.g. until it activates an end limit switch. This can be observed when switching on an inkjet printer; the controller will move the ink jet carrier to the extreme left and right to establish the end positions. A servomotor will immediately turn to whatever angle the controller instructs it to, regardless of the initial position at power up.

3.5 TELEMETRY PAIR:

Telemetry is an automated communications process by which measurements and other data are collected at remote or inaccessible points and transmitted to receiving equipment for monitoring.

A telemeter is a device used to remotely measure any quantity. It consists of a sensor, a transmission path, and a display, recording, or control device. Telemeters are the physical devices used in telemetry. Electronic devices are widely used in telemetry and can be wireless or hard-wired, analog or digital.

This telemetry pair is also known as trans-receiver pair because any one can be used as transmitter or receiver. These pairs are paired to each other at a certain radio frequency. These are categorized by their Radio frequency which decides the range and type of transmission. The transmitter will be connected to the Arduino board at serial communication ports and receiver will be connected to computer. To display the received data, I used HYPERTERMINAL which is a software designed for different versions of windows. The result can also be visualized from serial monitor.

3.6 RELAY:

A **relay** is an electrically operated switch. Many relays use an electromagnet to mechanically operate a switch, but other operating principles are also used, such as solid-state relays. Relays are used where it is necessary to control a circuit by a separate low-power signal, or where several circuits must be controlled by one signal. The first relays were used in long distance telegraph circuits as amplifiers: they repeated the signal coming in from one circuit and re-transmitted it on another circuit. Relays were used extensively in telephone exchanges and early computers to perform logical operations.

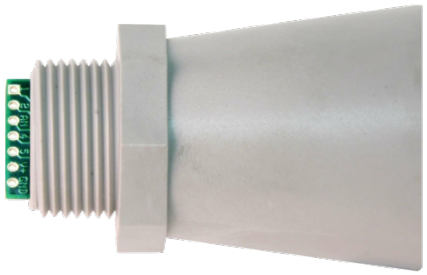
A type of relay that can handle the high power required to directly control an electric motor or other loads is called a contactor. Solid-state relays control power circuits with no moving parts, instead using a semiconductor device to perform switching. Relays with calibrated operating characteristics and sometimes multiple operating coils are used to protect electrical circuits from

overload or faults; in modern electric power systems these functions are performed by digital instruments still called "protective relays".

Magnetic latching relays require one pulse of coil power to move their contacts in one direction, and another, redirected pulse to move them back. Repeated pulses from the same input have no effect. Magnetic latching relays are useful in applications where interrupted power should not be able to transition the contacts.

Magnetic latching relays can have either single or dual coils. On a single coil device, the relay will operate in one direction when power is applied with one polarity, and will reset when the polarity is reversed. On a dual coil device, when polarized voltage is applied to the reset coil the contacts will transition. AC controlled magnetic latch relays have single coils that employ steering diodes to differentiate between operate and reset commands.

3.7 XL-MAXSONAR-WRA:



The XL-MaxSonar-WRA is our most popular weather resistant sensor designed for outdoor detection and ranging.

The weather resistant XL-MaxSonar-WR is a rugged ultrasonic sensor component module. This outdoor sensor provides very short to long distance detection and ranging in a compact, robust PVC housing. The ultrasonic sensor meets the IP67 water intrusion standard and matches standard electrical 3/4-inch PVC pipe fittings.

High output acoustic power combined with continuously variable gain, real-time background automatic calibration, real-time waveform signature analysis, and noise rejection algorithms results in virtually noise free distance readings. This holds true even in the presence of many of the various acoustic or electrical noise sources. The XL-MaxSonar-WR sensors are factory calibrated to match narrow sensor beam patterns and provide reliable long range detection zones.

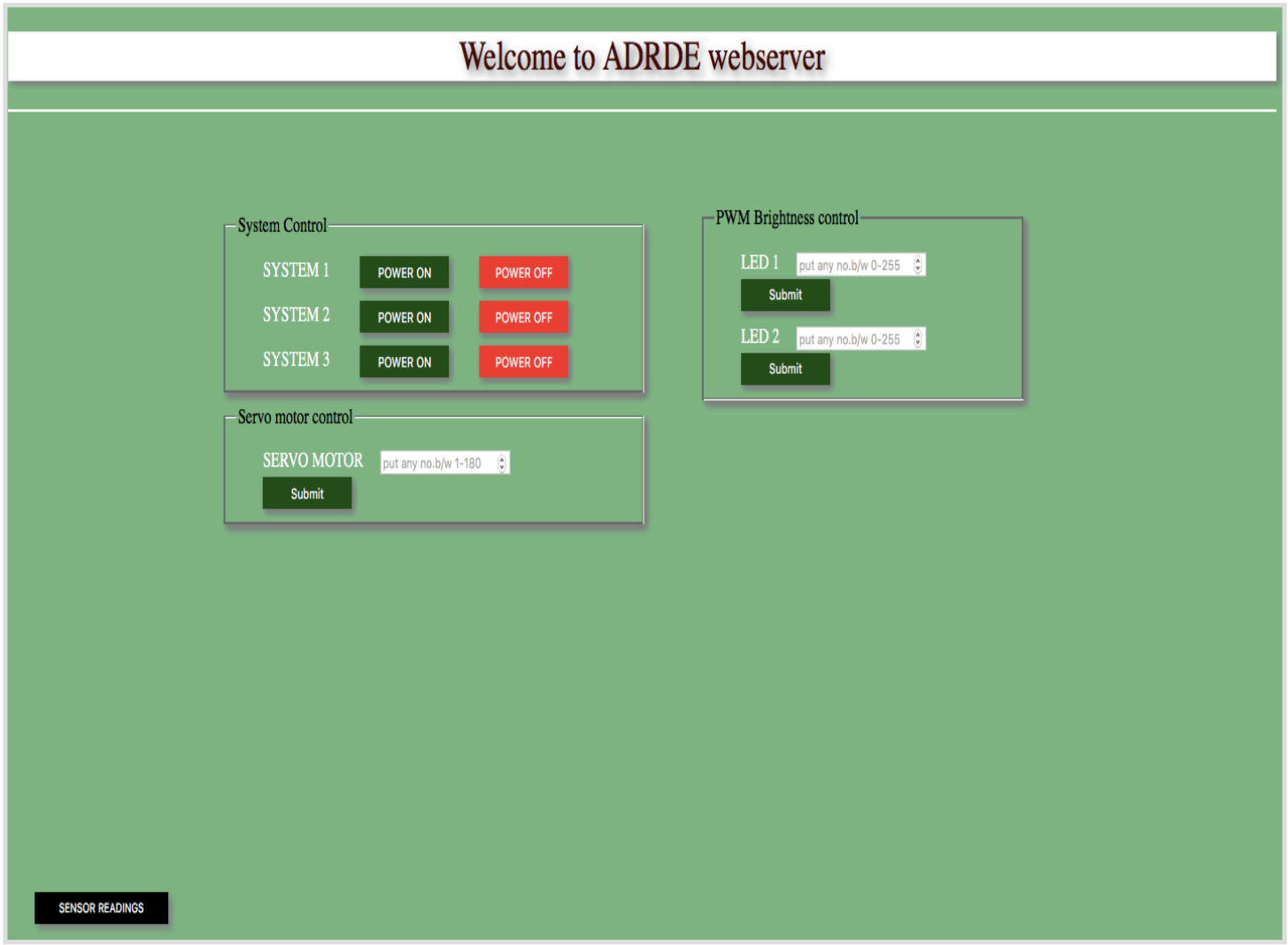
4. LOCAL AREA NETWORK BASED WEBSERVER FOR SYSTEM CONTROL:

4.1 Introduction:

This web server is used to control any electronic system which is attached to the board using LAN by just accessing the IP Address of the board. The web page (format: index.htm) is first stored in the SD Card which is on the Ethernet Shield and then the program which is used to control this web page is dumped on the board. The request which the web browser (Client here) gives can be seen on the Serial monitor by just printing the client requests in the program. We can make the page to auto refresh after a certain interval if we have to get any data from the analog pins otherwise there is no need to refresh the web page.

4.2 Web Page:

The web page is written using HTML, CSS. This gives the user an easy way to access the board.



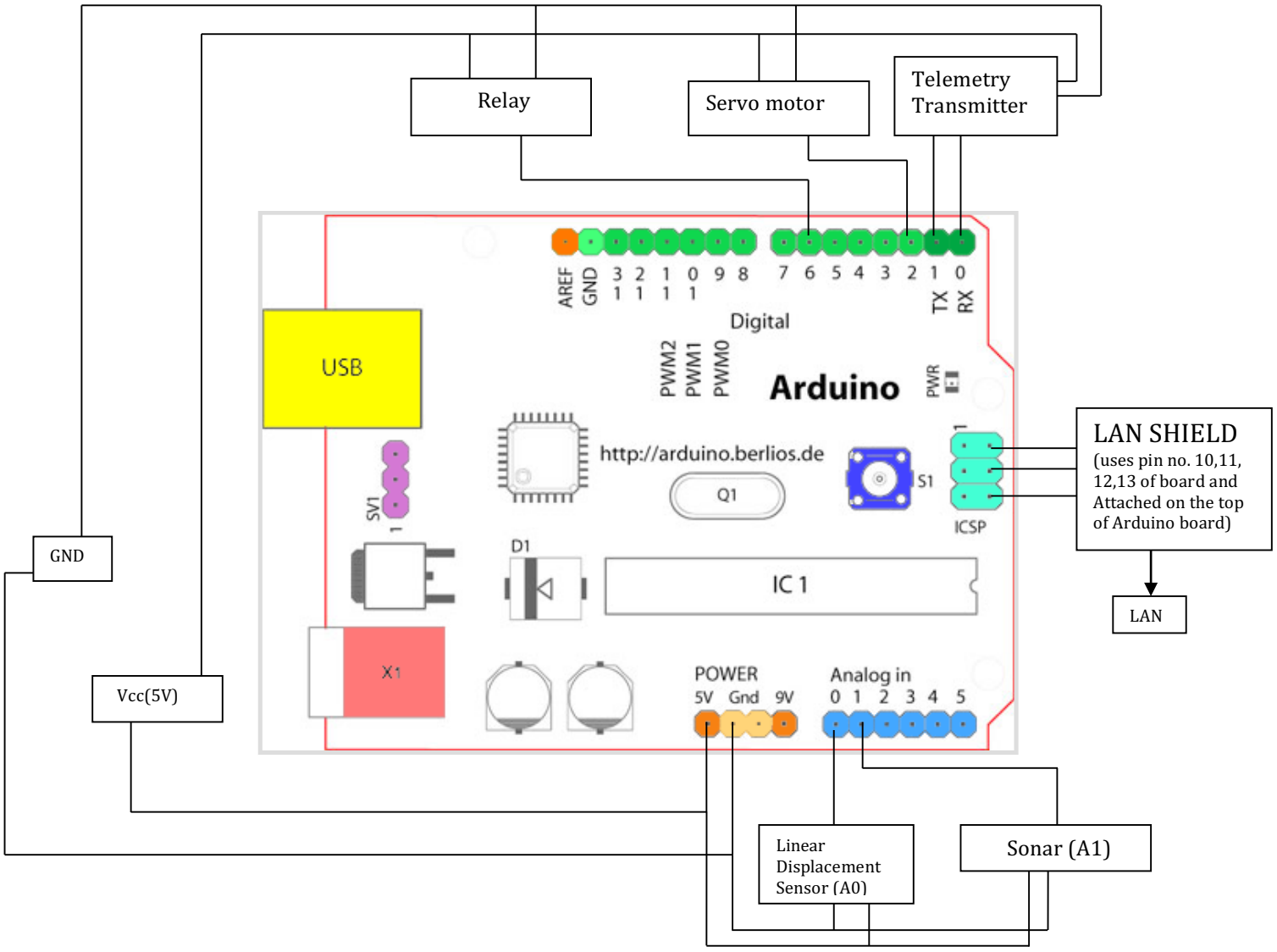
WEBPAGE CODE:

```
<!DOCTYPE HTML>
<html>
<head>
<title>Demo LED control</title>
<style>
 body{
        background-color: rgba(73, 153, 82, 0.7);
      }
.header1{
        color: #460000;
        text-shadow: 4px 4px 8px dimgrey;
        text-align: center;
        font-family: Broadway;
        font-weight: lighter;
        border: 1px;
        border-style: ridge;
        background-color: #ffffff;
        box-shadow: 3px 3px 5px grey;
                }
.tab1{
        color: #000000;
        margin-left: 15%;
        margin-top: 5%;
                }
.tab12{
        float: left;
         width: 100%;
         margin-left: 5%;
         margin-right: 5%;
         }
 .tab12n{
        float: left;
        width: 90%;
        padding-left: 20%;

      }
 .tab12s{
        float: left;
        width: 100%;
        padding-left: 5%;
}
  .down1{
        margin-top: 20%;
        float: left;
      }

.buttonof{
        align-items: center;}
 .b{
        margin-left: 30px;
        width: 100px;
        height: 25px;
        color: #ffffff;
        border: none;
        transition: 0.3s;
        cursor: pointer;
        box-shadow: 2px 2px 6px grey,3px 3px 5px
grey; }
```

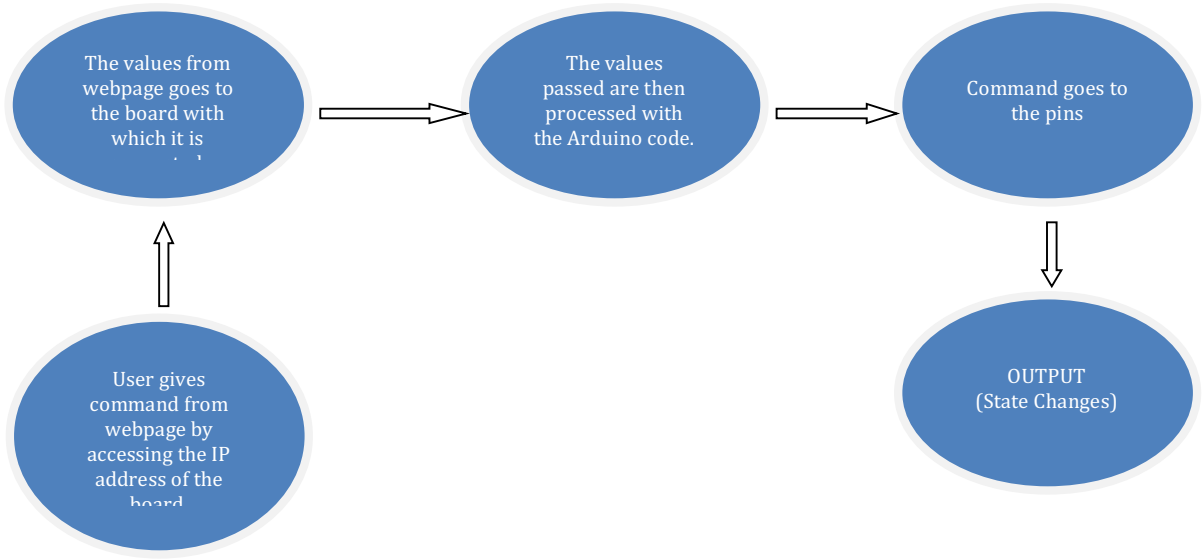
```
.b4{
        margin-left: 30px;
        width: 150px;
        height: 25px;
        border: none;
        transition: 0.3s;
        cursor: pointer;
      }
.b1{
          background-color: #004d1a;

                }
.b2{
          background-color: #ff0000;
                }
 .b3{
        background-color:  #000000;
        text-decoration: none;
        color: #ffffff;
        font-size: 10px;
        box-shadow: 4px 4px 6px grey;
      }
.b1:hover
                 {
          background-color: rgb(200,20,25);
          color: #ffffff;
          text-transform: lowercase;
          animation: ease-out;
                }
.b2:hover{
        background-color: rgb(200,20,25);
        color: #ffffff;
        text-transform: lowercase;
        animation: ease-out;
                }
.bol{
        color: #ffffff;
        padding-left: 30px;
        font-family: Perpetua;
        font-weight: lighter;
      }
.ba{
        margin-top: 10px;
      }
 button{
        box-shadow: white;
      }
 fieldset{
        box-shadow: 4px 4px 6px grey,3px 3px 5px
grey;
      }
          </style>
</head>
```

```
<body>
<div><h1 class="header1">Welcome to ADRDE webserver</h1>
          <hr color="#fffff"/>
          <form method="GET">
                    <table class="tab1"><tr><td>
        <div class="tab12">
          <fieldset>
            <legend>
              <p>System  Control</p>
            </legend>
            <div class="buttonof ba">
              <b class="bol">SYSTEM 1</b>
              <button name="LedOn1" class="b b1" type="submit" value="1">POWER ON</button>
              <button name="LedOff1" class="b b2" type="submit" value="2">POWER OFF</button>
            </div>
            <div class="buttonof ba">
              <b class="bol">SYSTEM 2</b>
              <button name="LedOn2" class="b b1" type="submit" value="3">POWER ON</button>
              <button name="LedOff2" class="b b2" type="submit" value="4">POWER OFF</button>
            </div>
            <div class="buttonof ba">
              <b class="bol">SYSTEM 3</b>
              <button name="LedOn3" class="b b1" type="submit" value="5">POWER ON</button>
              <button name="LedOff3" class="b b2" type="submit" value="6">POWER OFF</button>
            </div></fieldset>
        </div>
      </td>
   <td>
      <div class="tab12n">
        <fieldset>
          <legend>
            <p> PWM Brightness control</p>
          </legend>
          <div class="buttonof ba">
            <b class="bol">LED 1 </b>
            <input type="number" name="led1" placeholder="put any no.b/w 0-255" style="margin-left: 15px"/>
            <button name="led2" class="b b1" type="submit" value="11">Submit</button>
          </div>
          <div class="buttonof ba">
            <b class="bol">LED 2 </b>
            <input type="number" name="led3" placeholder="put any no.b/w 0-255" style="margin-left: 15px"/>
            <button name="led4" class="b b1" type="submit" value="12">Submit</button>
          </div>
        </fieldset> </div>
   </td> </tr>
<tr>  <td>  <div class="tab12s">
                <fieldset><legend>
                    <p>Servo motor control</p>
                  </legend>
                  <div class="buttonof ba">
                    <b class="bol">SERVO MOTOR</b>
                    <input type="number" name="serv1" placeholder="put any no.b/w 1-180" style="margin-left: 15px"/>
                    <button name="serv2" class="b b1" type="submit" value="112">Submit</button>
                  </div> </fieldset> </div></td></tr>
</table>  </form></div>
<div class="down1">
   <button type="submit" onclick="newpop()" class="b4 b3" >SENSOR READINGS</button>
</div></body>
</html>
```

4.3 CIRCUIT DIAGRAM:



4.4 CODE PROCESSING:



4.5 Connecting LAN Shield with Arduino UNO and accessing SD card files:

Arduino Code:

```
#include <Servo.h>              //include the Servo Library for using Servo with Arduino
#include <SD.h>                 //include the SD Card Library for accessing SD card from LAN Shield
#include <SPI.h>               //include Serial Library for using serial communication
#include <Ethernet.h>         //include Ethernet Library for using Ethernet Shield

String HTTP_req;
EthernetClient client; //Initialize Arduino Ethernet Client
String readString;

char fade=0;       //Initializing the variable fade to 0
int pos=0;       //Initializing the variable pos to 0
int servoPin=2;        //Setting up Pin no. with which we want to use Servo Library
int servoDelay=40  //Setting up Servo Delay

Servo myPointer;   //Making the Object of Servo for using the functions of Servo Library

// Enter a MAC address and IP address for your controller below.
byte mac[] = {
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED
};

IPAddress ip(192, 168, 0, 5);   // The IP address will be dependent on your local network
EthernetServer server(80);     //Generally Ethernet uses port no. 80
File WebFile;

void setup() {     //Setting up the pinMode to Output
  pinMode(6,OUTPUT);
  pinMode(2,OUTPUT);
  // Open serial communications and wait for port to open:
  Serial.begin(9600);     // Baud rate must be same as the baud rate of Serial Monitor
  Ethernet.begin(mac, ip);      // start the Ethernet connection and the server:
  server.begin();
  Serial.print("server is at ");
  Serial.println(Ethernet.localIP());
   Serial.println("Initializing SD card.....");      //Initializing SD card
  if(!SD.begin(4)){
    Serial.println("Error-  SD card not found");
    return;}
Serial.println("SUCCESS- SD card initialized...");
if(!SD.exists("index.htm")){   //check for index.htm file
  Serial.println("ERROR - index file not found");
    return;}
Serial.println("SUCCESS - index file found..");
myPointer.attach(servoPin);}


void loop() {
  //listen for incoming clients
  EthernetClient client = server.available();
  if (client) {
    Serial.println("new client");
    boolean currentLineIsBlank = true;          // an http request ends with a blank line
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        Serial.write(c);
       if(readString.length() < 100){     //read char by char HTTP request
         readString += c;         //Store character to string
       }
          if(c=='\n' && currentLineIsBlank){
```

```
            client.println ("HTTP/1.1 200 OK ");
            client.println ("Content-Type: text/html");
            client.println ("Connection : close ");
            client.println ( );
                        WebFile = SD.open("index.htm");     // open webfile index.htm from sd card
          if(WebFile){
            while(WebFile.available()){
              client.write (WebFile.read());
            }
            WebFile.close();
          }
         if (c == '\n') {                  // if HTTP req has ended check
           Serial.println(readString); //print to serial monitor for debugging
           currentLineIsBlank = true; // you're starting a new line
         }
            else if (c != '\r') {
           currentLineIsBlank = false;              // you've gotten a character on the current line
         }
                delay(1);     // give the web browser time to receive the data
        client.stop();           // close the connection:
        Serial.println("client disconnected");

        // control the arduino pins

 if(readString.indexOf("LedOn1")>0)  //Switching ON the LED if ON Button Pressed
        {
          digitalWrite(6,HIGH);
          Serial.println("Led ON");
        }
        if(readString.indexOf("LedOff1")>0)  //Switching OFF the LED if OFF Button Pressed
        {
          digitalWrite(6,LOW);
          Serial.println("LED OFF");
        }


if(readString.indexOf("serv2")>0)
        {
         //pos = readString.indexOf("serv1");   //Starting the servo to sweep from one angle to another
        while(true){
        for(int i=10; i<=180 ;i=i+5){
           myPointer.write(i);
           Serial.println(i);
            delay(servoDelay); }
          for(int j=180; j>=10 ;j=j-5){
           myPointer.write(j);
           Serial.println(j);
            delay(servoDelay); }}

          if(readString.indexOf("led2")>0){         //changing the brightness of LED (Fading)
           fade = int(readString.indexOf("led1"));
            Serial.println(fade);
             for(int i=0; i<=255; i=i+5){
               analogWrite(6,i);
               delay(40); }
             for(int j=255; j>=0; j=j-5){
             analogWrite(6,j);
             delay(40);
             } }
         //clear readString for next read
         readString="";
}
}
 }
}
```

5.SERIAL COMMUNICATION

5.1 WHAT IS SERIAL COMMUNICATION?

In télécommunication and data transmission, **serial communication** is the process of sending data one bit at a time, sequentially, over a communication Channel or computer bus. This is in contrast to parallel communication, where several bits are sent as a whole, on a link with several parallel channels.
Serial communication is used for all long-haul communication and most computer networks, where the cost of cable and synchronization difficulties make parallel communication impractical. Serial computer buses are becoming more common even at shorter distances, as improved signal integrity and transmission speeds in newer serial technologies have begun to outweigh the parallel bus's advantage of simplicity (no need for serializer and deserializer, or SerDes) and to outstrip its disadvantages (clock skew, interconnect density). The migration from PCI to PCI Express is an example.

Serial in Arduino programming is used for communication between the Arduino board and a computer or other devices. All Arduino boards have at least one serial port (also known as a UART or USART): Serial. It communicates on digital pins 0 (RX) and 1 (TX) as well as with the computer via USB. Thus if you use these functions, you cannot also use pins 0 and 1 for digital input or output.

You can use the Arduino environment's built-in serial monitor to communicate with an Arduino board. Click the serial monitor button in the toolbar and select the same baud rate used in the call to `begin()`.

Serial communication on pins TX/RX uses TTL logic levels (5V or 3.3V depending on the board). Don't connect these pins directly to an RS232 serial port; they operate at +/- 12V and can damage your Arduino board.

5.2 HYPERTERMINAL:

HyperTerminal is the windows terminal emulation program capable of connecting to systems through TCP/IP Networks, Dial-Up Modems, and COM ports.

1. Open HyperTerminal:
2. Enter a name for the new connection and select an icon, and then click OK. The Connect To box opens.
3. Select the PC COM port that connects the PC to the controller, and then click OK.
4. The Port Settings box opens as shown below. To enable your PC to communicate with the controller, set the COM port parameters to the M90 default settings: BPS 9600, Data bits=8, Parity=N, Stop bits=1, Flow control=none, and then click OK and in properties select the correct port no. with which the receiver is connected.

6. CLOUD BASED DATA MONITORING

6.1 Introduction to Cloud Platform:

For the visualization of data coming from Arduino Board we use THINGSPEAK, a Cloud platform powered by MathWorks. This platform gives you an easy way to plot the data on its server. It is the widest IOT platform which is being used worldwide for data analysis of IOT devices. It supports MATLAB data analysis. It gives a nice user-interface (UI) for the visualization of data.
On THINSPEAK server you can make your own account and then you will be given so many channels features. Each user will be assigned a certain API key, which is specific. On writing the code in Arduino you have to specify the API Key and channel name where you want to plot your incoming analog data.

6.2 ALGORITHM:

- Connect the Sensor to the pins shown in the Circuit Diagram.
- Include the ThingSpeak library in addition with other libraries to connect with ThingSpeak cloud.
- Enter the mac address and then make the settings to connect to ThingSpeak server.
- In settings for ThingSpeak write correct API key and channel settings which you have set on the ThingSpeak Account.
- Initialize the Serial monitor for debugging and then start the Ethernet Client.
- Read the Analog Value from the sensor and store it in a string.
- Check the Ethernet connection with the client, if connected then update the ThingSpeak fields.
- Then Send the data to ThingSpeak server using standard HTTP protocol.

6.3 ARDUINO CODE:

```
#include <SPI.h>     //Include the Serial communication Library
#include <Ethernet.h> //Include Ethernet Library
#include <ThingSpeak.h> //Include ThingSpeak Library
// Local Network Settings
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED }; // Must be unique on local network
// ThingSpeak Settings
char thingSpeakAddress[] = "api.thingspeak.com";
String writeAPIKey = "XXXMX2WYYR0EVZZZ"; //Specific API key is provided to each user on SignUp on
ThinSpeak Server
const int updateThingSpeakInterval = 16 * 1000;      // Time interval in milliseconds to update
ThingSpeak (number of seconds * 1000 = interval)
// Variable Setup
long lastConnectionTime = 0;
boolean lastConnected = false;
```

```
int failedCounter = 0;

// Initialize Arduino Ethernet Client
EthernetClient client;

void setup()
{
  // Start Serial for debugging on the Serial Monitor
  Serial.begin(9600);
  // Start Ethernet on Arduino
  startEthernet();
}
void loop()
{
  // Read value from Analog Input Pin 0 where Sensor is connected
  String analogValue0 = String(analogRead(A0), DEC);
  // Print Update Response to Serial Monitor for easy debugging
  if (client.available())
  {
    char c = client.read();
    Serial.print(c);
  }
  // Disconnect from ThingSpeak
  if (!client.connected() && lastConnected)
  {
    Serial.println("...disconnected");
    Serial.println();
    client.stop();
  }

  // Update ThingSpeak
  if(!client.connected() && (millis() - lastConnectionTime > updateThingSpeakInterval))
  {
    updateThingSpeak("field1="+analogValue0);
  }
  // Check if Arduino Ethernet needs to be restarted
  if (failedCounter > 3 ) {startEthernet();}
  lastConnected = client.connected();
}
void updateThingSpeak(String tsData)
{
  if (client.connect(thingSpeakAddress, 80))
  {
    client.print("POST /update HTTP/1.1\n");
    client.print("Host: api.thingspeak.com\n");
    client.print("Connection: close\n");
```

```
    client.print("X-THINGSPEAKAPIKEY: "+writeAPIKey+"\n");
    client.print("Content-Type: application/x-www-form-urlencoded\n");
    client.print("Content-Length: ");
    client.print(tsData.length());
    client.print("\n\n");
    client.print(tsData);

    lastConnectionTime = millis();
    if (client.connected())
    {
      Serial.println("Connecting to ThingSpeak...");
      Serial.println();
      failedCounter = 0;
    }
    else
    {
      failedCounter++;
      Serial.println("Connection to ThingSpeak failed ("+String(failedCounter, DEC)+")");
      Serial.println();
    }  }
  else
  {
    failedCounter++;
    Serial.println("Connection to ThingSpeak Failed ("+String(failedCounter, DEC)+")");
    Serial.println();
    lastConnectionTime = millis();
  } }
void startEthernet()
{

 client.stop();
  Serial.println("Connecting Arduino to network...");
  Serial.println();
  delay(1000);
  // Connect to network amd obtain an IP address using DHCP
  if (Ethernet.begin(mac) == 0)
  {
    Serial.println("DHCP Failed, reset Arduino to try again");
    Serial.println();
  }
  else
  {
    Serial.println("Arduino connected to network using DHCP");
    Serial.println();
  }
  delay(1000);
```

7. CONCLUSION

During the course of training at ADRDE I worked on the Application of Internet Of Things (IOT). The main aim of the project was to make a LAN based webpage to control different systems with Arduino Uno and data monitoring with a cloud platform, ThingSpeak powered by Mathworks which uses Matlab Data analysis tools for visualization of data. This cloud platform is widely used worldwide for IOT. The project was completed and the my next goal will be smart home automation system with Arduino Uno which will use online platform IFTTT(If This Then That) which helps in making the behavior of things smart meaning if something happens with the system then it will trigger the solution by itself and notify us on any social networking platform.
This project I was working on during training gave me a good learning experience of Arduino programming language and this will help me achieve my next goal.

8. REFERENCES

1. https://components101.com/microcontrollers/arduino-uno
2. https://www.arduino.cc/en/Reference/Board
3. https://en.wikipedia.org/wiki/Servomotor
4. https://en.wikipedia.org/wiki/Serial_communication
5. https://thingspeak.com/
6. https://www.arduino.cc/
7. https://www.datasciencebe.com/
8. http://www.nothans.com/