MODULE 3

Prototyping and Designing Software for IoT Applications: Introduction, Prototyping Embedded device software, Programming Embedded Device Arduino Platform using IDE, Reading data from sensors and devices, Devices, Gateways, Internet and Web/Cloud services software development.

Programming MQTT clients and MQTT server: Introduction to IoT privacy and security. Vulnerabilities, security requirements and threat analysis, IoT Security Tomography and layered attacker model.

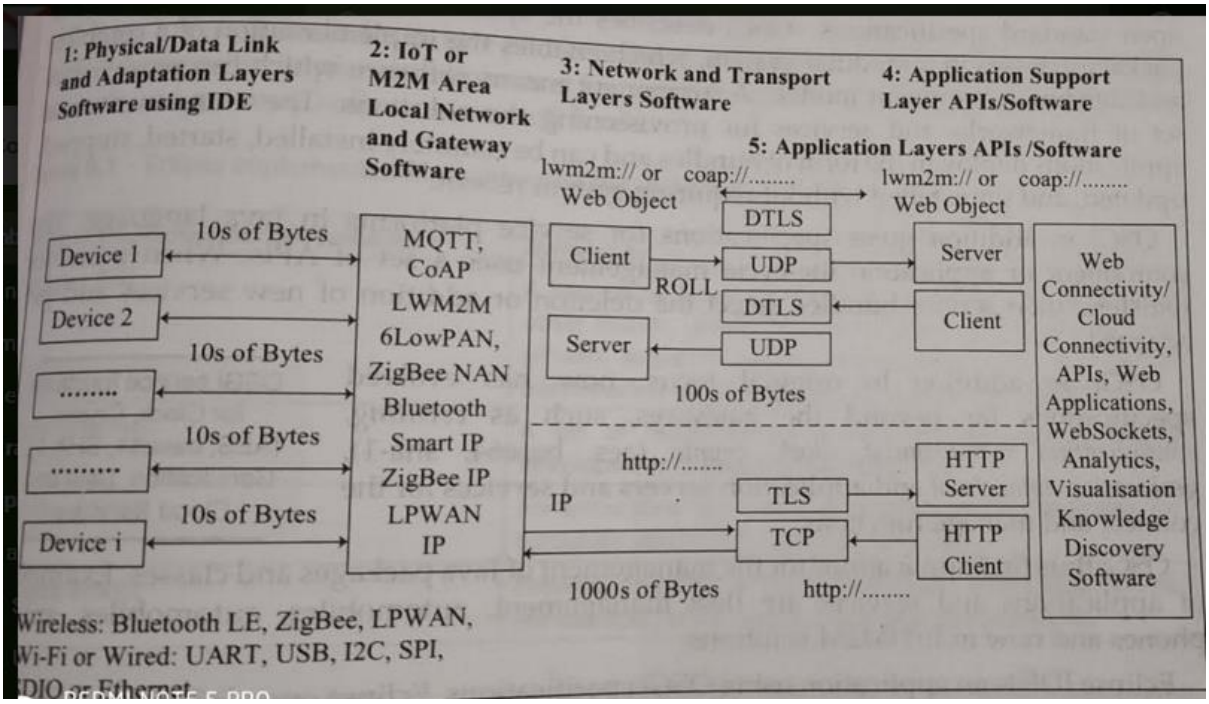
- This chapter focuses on methods of developing software, which are used at levels of IoT devices, gateways, Internet connectivity and web cloud and applications.
- **Explain Prototyping Embedded device software.**
  - Prototyping development of the programs requires bootloader, OS and IDE.
  - Software embeds into a device platform.
  - An IDE enables development of software for functions of data gathering, consolidation, and connection to Internet.
  - The IDE may also enables the use of OS or RTOS functions at an embedded device.
  - Bootloader stores at flash/ROM of a microcontroller device and enables communication with a computer having an IDE.
  - IDE has APIs, libraries,compilers, RTOS,simulator,editor, assembler, debugger, emulators, logic analyzer, code burner.
  - IDE enables the development of codes on a computer and downloads the codes on to an embedded device.
  - The code burner places codes into flash memory or EEPROM o EPROM.
  - Hence the application code is embedded into the device.
- **Discuss about Programming Embedded Devices Arduino Platform using IDE.**
  - Arduino is an open source tool for developing computers that can sense and control more of the physical world
  - It is an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board.
  - The Arduino development board is an implementation of wiring, a similar physical computing platform, which is based on the processing multimedia programming environment.
  - The arduino board can be programmed using avr-gcc tools.

- The board has a pre installed bootloader to perform the operation.
- It provisions or multitasking by the usage of interrupt handling functions for each task.
- The communication between the computer and the arduino is 'serial' , because data is broken down into bits, each sent one after the other down a single wire.
- The Arduino board can communicate at various "baud rates".
- A baud is a measure of how many times the hardware can send 0's and 1's in a second
- The baud rate must be set properly for the board to convert incoming and outgoing information to useful data.
- If your receiver is expecting to communicate at a baud rate of 2400, but your transmitter is transmitting at a different rate, the data you get will not make sense.
- To set the baud rate, use the following code:
  void setup() { Serial.begin(9600); }
- Other standard baud rates available on most Arduino modules include: 300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, or 115200.
- Arduino software interface:The software used by the arduino is Arduino IDE.
- The open-source Arduino Software (IDE-Integrated development Environment) makes it easy to write code and upload it to the board.
- IDE runs on Windows, Mac OS X, and Linux. The environment is written in Java.
- It is designed to introduce programming to artists and other newcomers unfamiliar with software development.
- It includes a code editor with features such as syntax highlighting, brace matching, and is also capable of compiling and uploading programs to the board with a single click.
- The code for the arduino is written in C/C++.
- A serial monitor at the IDE enables messages from the embedded software for the microcontroller set up into the computer screen.
- The messages are required during testing and debugging the downloaded software during test stages.

- **Write note on Emulator and debugger.**
  - Emulating and debugging of software ensures reliable, tested and bug free software.
  - Emulation means to do actions similar to the real entity.
  - Emulator software emulates the running of an embedded device platform known as target board.
  - Software emulates the embedded device platform onto the computer.
  - In circuit emulation (ICE) means hardware debugging by emulation after connecting the target board to the computer.

- The computer sets the break points in the embedded software to help the debugging of the code.

- **Discuss about the devices, gateways, internet and web/ cloud services software development**



- The IOT / M2M devices use CoAP and LWM2M web communication protocols and message communication protocols such as message cache, MQTT and XMPP.
- They use the communication and communicate over the web gateway.
- An Internet of Things (IoT) gateway is a physical device or software program that serves as the connection point between the cloud and controllers, sensors and intelligent devices.
- All data moving to the cloud, or vice versa, goes through the gateway, which can be either a dedicated hardware appliance or software program.
- An IoT gateway may also be referred to as an intelligent gateway or a control tier.
- A gateway provides a place to preprocess that data locally at the edge before sending it on to the cloud.
- When data is aggregated, summarized and tactically analyzed at the edge, it minimizes the volume of data that needs to be

forwarded on to the cloud, which can have a big impact on response times and network transmission costs.

- ❖ Another benefit of an IoT gateway is that it can provide additional security for the IoT network and the data it transports.
- ❖ Because the gateway manages information moving in both directions, it can protect data moving to the cloud from leaks and IoT devices from being compromised by malicious outside attacks with features such as tamper detection, encryption, hardware random number generators and crypto engines.

- **Explain the requirements of privacy and security vulnerabilities from threats.**
  - ❖ Message privacy means no interference or disturbance inn transmission of the message.
  - ❖ The following could be done to improve privacy, threat analysis in the IoT devices while communication:

1. **Minimize data acquisition**
2. **Minimize the number of data sources**
3. **Minimize raw data intake**
4. **Minimize knowledge discovery**
   IoT applications should discover only the knowledge necessary to achieve their primary objectives. For example, if the objective is to recommend food plans, the app should not attempt to infer users' health status without their explicit permission.
5. **Minimize data storage**
   Raw data should be deleted once secondary context is derived.
6. **Encrypt data communications**
   Typically, device-to-device communications are encrypted at the link layer using special electronic hardware included in the radio modules. Gateway-to-cloud communication is typically secured through HTTPS using Secure Sockets Layer (SSL) or Transport Layer Security (TLS).
7. **Encrypt data during processing**
8. **Reduce data granularity**
   IoT applications should request the minimum level of granularity that is required to perform their primary tasks. A higher level of granularity could lead to secondary data usage and eventually privacy violations
9. **Distribute data processing**
   Distributed data processing avoids centralized large-scale data gathering and exfiltration

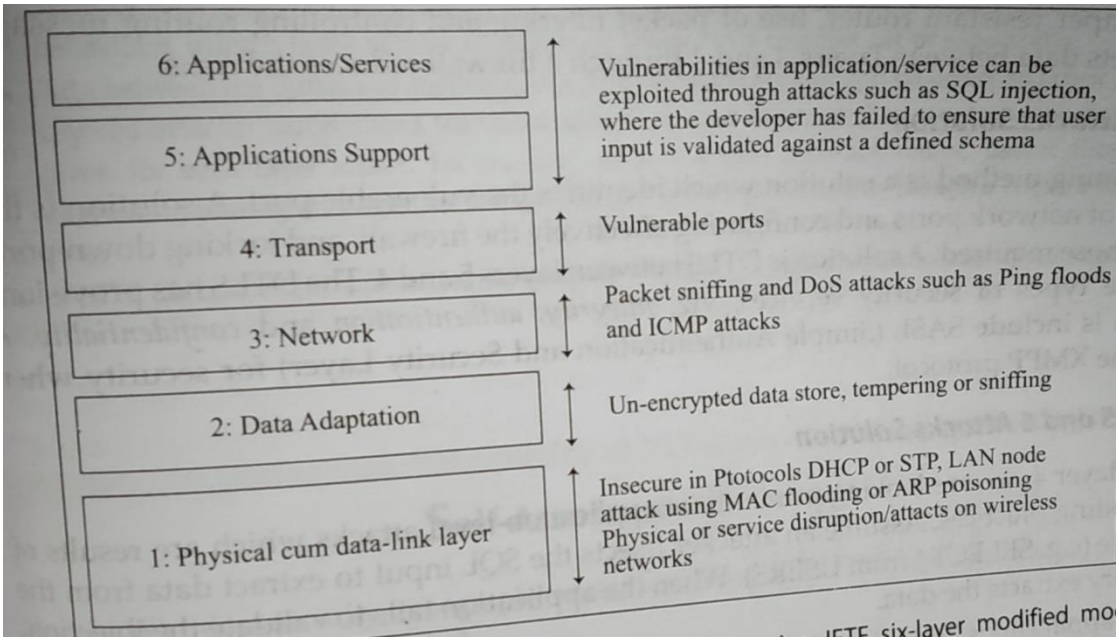
Software attacks and Network Attacks

- **Physical attacks**: These types of attacks tamper with the hardware components and are relatively harder to perform because they requires an expensive material.
- Some examples are de-packaging of chip, layout reconstruction, micro-probing, particle beam techniques, etc.
- **Side channel attacks:** These attacks are based on a side channel Information that can be retrieved from the encryption device that is neither the plaintext to be encrypted nor the cipher text resulting from the encryption process.
- Encryption devices produce timing information that is easily measurable.
- **Cryptanalysis attacks:** These attacks are focused on the ciphertext and they try to break the encryption, i.e. find the encryption key to obtain the plaintext.
- Examples of cryptanalysis attacks include ciphertext only attack, known-plaintext attack, chosen-plaintext attack, man-in-the-middle attack, etc.
- **Software attacks**: Software attacks are the major source of security vulnerabilities in any system.
- Software attacks exploit implementation vulnerabilities in the system through its own communication interface.
- This kind of attack includes exploiting buffer overflows and using Trojan horse programs, worms or viruses to deliberately inject malicious code into the system.
- Jamming attack is the one of the ruinous invasion which blocks the channel by introducing larger amount of noise packets in a network.
- Jamming is the biggest threat to IoT where a network consists of small nodes with limited energy and computing resources.
- So it is very difficult to adopt the conventional anti jamming methods to implement over IoT technologies.
- **Network Attacks:** Wireless communications systems are vulnerable to network security attacks due to the broadcast nature of the transmission medium.
- Basically attacks are classified as active and passive attacks. Examples of passive attacks include monitor and eavesdropping, Traffic analysis, camouflage adversaries, etc.
- Examples of active attacks include denial of service attacks, node subversion, node malfunction, node capture, node outage, message corruption, false node, and routing attacks, etc
-

- **Explain IoT Security Tomography and layered attacker model**

Or

- **Outline the security tomography of large networks and the layered attacker model.**
  - Computational tomography means a computing method of producing a 3D picture of the internal structure of an object by observation and recording the differences in the effects of passage of energy waves.
  - Computational security is a complex set of networks utilises the network tomography procedures of identifying the network vulnerabilities.
  - The layered attacker model is as shown below:



The following are the solutions for the mitigating the attacks:

Layer 1 solution:

The devices used decide the type of security required.

Layer 2 solution:

Programming the network switches to prevent internal node attacks during the use of DHCP or spanning tree protocol. Additional controls may includeARP inspection,disabling the unused ports and enforcing effective security on VLANs.

Layer 3 attack solution:

Use a temper resistant router, use of packet filtering and controlling routing messages and packets data between the layers 3 and 4 through the firewalls reduces the risk.

Layer 4 attack solution:
Port scanning method is a solution which identifies the vulnerable port.
Layer 5 attack solution:
Application level attacks are a resultant of poor coding practices.web applications/services can use HTTPS communication link.

Programs:

The Arduino Integrated Development Environment (IDE) is a cross-platform application (for Windows, macOS, Linux) that is written in functions from C and C++[2]. It is used to write and upload programs to Arduino compatible boards.

The Arduino IDE supports the languages C and C++ using special rules of code structuring.[5]

The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures

It is enough to connect it to the computer USB port and press the "Upload" icon to start a process that transfers your sketch into the Flash memory of the microcontroller.

This transfer described above happens thanks to a special piece of code that is executed at every reset of the microcontroller and that looks for a sketch to be uploaded from the serial/USB port using a specific protocol and speed. If no connection is detected, the execution is passed to the code of your sketch.

This little (usually 512 bytes) piece of code is called the "Bootloader" and it is in an area of the memory of the microcontroller – at the end of the address space - that can't be reprogrammed as a regular sketch and had been designed for such purpose.

Assume Arduin0 Uno board as an embedded device platform for the following project.

Three traffic lights red, yellow and Green must be controlled on each of the four north, south, east and west pathways.

North and south should turn green

East and west should turn red.

```
/*  North  - ledR0,ledY0, ledG0
East – ledR1,ledY1, ledG1
South -  ledR2,ledY2, ledG2
West -  ledR3,ledY3, ledG3  */
intinternalLED=13;
ledR0 = 2;    …..ledG3 =14;   /* Assign pin no to all leds*/
voidnorth_south_green()
```

```
{
        digitalWrite(ledR0,LOW);
        digitalWrite(ledY0,LOW);
        digitalWrite(ledG0,HIGH);
        digitalWrite(ledR2,LOW);
        digitalWrite(ledY2,LOW);
        digitalWrite(ledG2,HIGH);
}
voideast_west_red()
{
        /*Write LED on off as above*/
}
Void setup()
{
    /*  GPIO pins  which are connected to  LED's are set as output */
    pinmode(ledR0, OUTPUT);
    /* Set for all led's */
    pinmode(internalLED, OUTPUT);
    digitalWrite(internalLED,HIGH);
    serial.begin(9600);
    serial.printlm("print any message");
}
Void loop()
{
    north_south_green();
    east_west_red();
}
```

Program 2 :Assume delays of 10s between successive states of LEDs and steady state for 30s for a pair of pathways.

```
/*   North  - ledR0,ledY0, ledG0
```

East – ledR1,ledY1, ledG1

South - ledR2,ledY2, ledG2

West - ledR3,ledY3, ledG3 */

intinternalLED=13;

ledR0 = 2; .....ledG3 =14; /* Assign pin no to all leds*/

```
voidnorth_south_green()
{
        digitalWrite(ledR0,LOW);
        digitalWrite(ledY0,LOW);
        digitalWrite(ledG0,HIGH);
        digitalWrite(ledR2,LOW);
        digitalWrite(ledY2,LOW);
        digitalWrite(ledG2,HIGH);
}
voidnorth_south_yellow();
void north_south_red(); /* write digiralWrite function as above*/
voideast_west_red()
voideast_west_yellow()
voideast_west_green()
Void setup()
{
    /* GPIO pins which are connected to LED's are set as output */
    pinmode(ledR0, OUTPUT);
    /* Set for all led's */
    pinmode(internalLED, OUTPUT);
    digitalWrite(internalLED,HIGH);
    serial.begin(9600);
```

```
        serial.printlm("print any message");
}
Void loop()
{
        north_south_green();
        east_west_red();
        delay(30000);
voidnorth_south_yellow();
voideast_west_yellow()
delay(10000);
        north_south_red();
        east_west_green();
        delay(30000);
}
```

Program 3: A temperature sensor analog output is given to an SPI at Arduini through a 10bit ADC and PISO. How will be the data read from SPI input every hour into an Arduino board from the sensor.

How will the one hour wait loop be programmed and how does the test performed by LED on on-off sate using a blinking program blink at each 3s interval.?

Let temp sensor range – 0-100 degree

Supply voltage -3.3V

ADC -10bit

Calib $=\frac{100}{1023}$ = .097752

```
#include <SPI.h>
#include<util.h>
#define tempreading = pin2 /* sensor name at pin2*/
#define calib= .097752
Float sensoranalogvalue;
Float calculated value;
```

Void setup()

```
{
Sensoranalogvalue = 0;
calculated value =0;
        pinmode(internalLED, OUTPUT);
        digitalWrite(internalLED,HIGH);
        serial.begin(9600);
        serial.printlm("print any message");
}
```

Void loop()

```
{
Sensoranalogvalue = analogRead(tempreading);  /* read from sensor*/
calculated value =calib * Sensoranalogvalue * 1023/3.3;
serial.print("temperature");
}
```

Void test() /* one hour loop with blinking of 3s on-off*/

```
{
        For(int i=1; i<= 600 ;i++)
        {
                digitalWrite(internalLED,HIGH);
                delay(3000);
                digitalWrite(internalLED,LOW);
                delay(3000);
        }
}
```