Assignment Number - 10

Title: Write a Client / Server application by using Raspberry-Pi.

Problem Definition: Write a server application to be deployed on Raspberry-Pi / BeagleBoard. Write client applications to get services from the server application.

Objectives:

- ➔ To understand functionalities of various single board embedded platform fundamentals.
- ➔ To develop client server applications.

Outcomes:

Students will be able to:

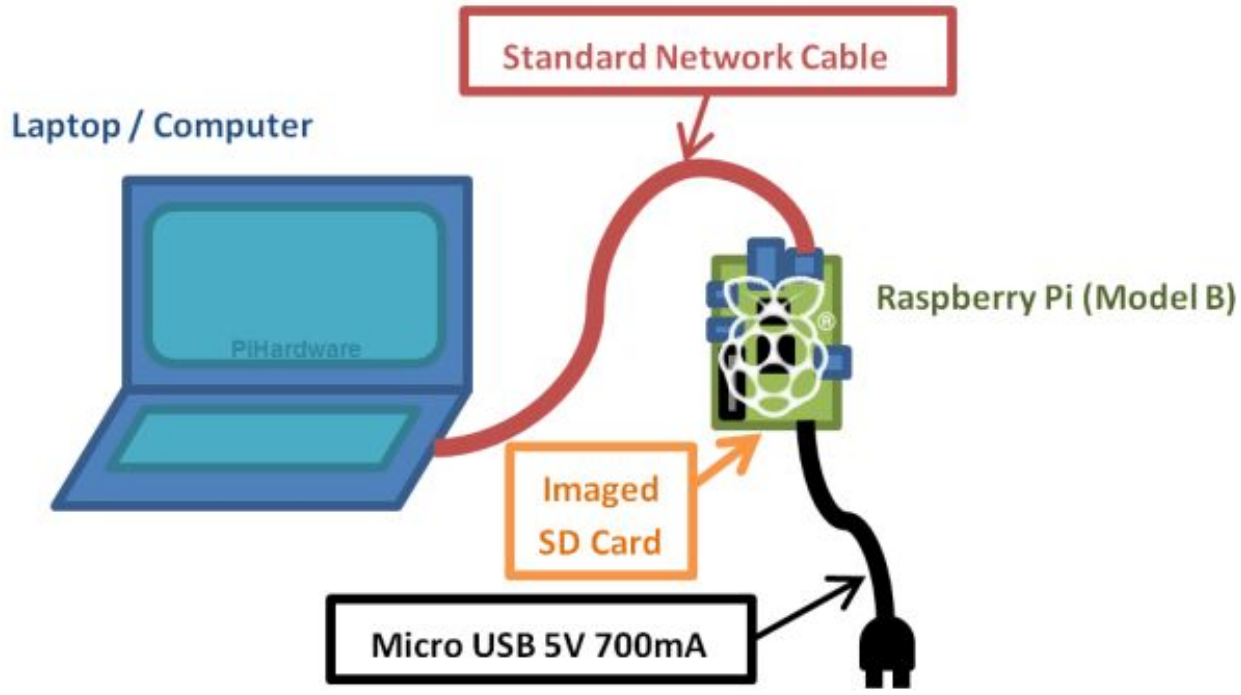
- ➔ Implement an architectural design for IoT for specified requirements.
- ➔ Solve the given societal challenge using IoT.

Software and Hardware Requirements:

Laptop / Computer, Raspberry-Pi, Ethernet Cable, Micro USB

Theory:

A simple example to get started. The Raspberry-Pi runs a server that waits for connection from a laptop, and expects integers from it. It multiples each integer by 2 and sends it back. The laptop runs a client that initiates a connection, sends a bunch of positive integers that it gets back multiplied by two, and closes the connection by sending a -1. Sending a -2 causes the server to stop.



Source Code:

Server Code

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
void error (char *msg)
{
        perror (msg);
        exit (1);
}
int func (int a)
{
        return 2*a;
}
void sendData (int sockfd, int x)
{
        int n;
        char buffer[32];
        sprintf (buffer, "%d\n", x);
        if ((n = write (sockfd, buffer, strlen (buffer))) < 0)
                error (const_cast<char*> ("ERROR writing to socket"));
        buffer[n] = '\0';
}
int getData (int sockfd)
{
        char buffer[32];
        int n;
        if ((n = read (sockfd, buffer, 31)) < 0)
                error (const_cast<char*> ("ERROR reading from socket"));
        buffer[n] = '\0';
        return atoi (buffer);
}
int main (int argc, char *argv[])
{
        int sockfd, newsockfd, portno = 51717, clilen;
        char buffer[256];
        struct sockaddr_in, serv_addr, cli_addr;
```

```
        int n;
        int data;
        printf ("using port #%d\n", portno);
        sockfd = socket (AF_INET, SOCK_STREAM, 0);
        if (sockfd < 0)
                error (const_cast<char*> ("ERROR opening socket"));
        bzero ((char*) &serv_addr, sizeof (serv_addr));
        serv_addr.sin_family = AF_INET;
        serv_addr.sin_addr.s_addr = INADDR_ANY;
        serv_addr.sin_port = htons (portno);
        if (bind (sockfd, (struct sockaddr*) &serv_addr, sizeof (serv_addr)) < 0)
                error (const_cast<char*> ("ERROR on binding"));
        listen (sockfd, 5);
        clilen = sizeof (cli_addr);
        while (1)
        {
                printf ("waiting for new client…\n");
                if ((newsockfd = accept (socfd, (struct sockaddr*) &cli_addr, (socklen_t*)
&clilen)) < 0)
                        error (const_cast<char*> ("ERROR on accept"));
                printf ("opened new communication with client\n");
                while (1)
                {
                        data = getData (newsockfd);
                        printf ("got %d\n", data);
                        if (data < 0)
                                break;
                        data = func (data);
                        printf ("sending back %d\n", data);
                        sendData (newsockfd, data);
                }
                close (newsockfd);
                if (data == -2)
                        break;
        }
        return 0;
}
```

Client Code

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <arpa/inet.h>
void error (char *msg)
{
        perror (msg);
        exit (0);
}
void sendData (int sockfd, int x)
{
        int n;
        char buffer[32];
        sprintf (buffer, "%d\n", x);
        if ((n = write (sockfd, buffer, strlen (buffer))) < 0)
                error ((const_cast<char*> ("ERROR writing to socket"));
        buffer[n] = '\0';
}
int getData (int sockfd)
{
        char buffer[32];
        int n;
        if ((n = read(sockfd, buffer, 31)) < 0)
                error (const_cast<char*> ("ERROR reading from socket"));
        buffer[n] = '\0';
        return atoi (buffer);
}
int main (int argc, char *argv[])
{
        int sockfd, portno = 51717, n;
        char serverIp[] = "169.254.0.2";
        struct sockaddr_in, serv_addr;
        struct hostent *server;
        char buffer[256];
        int data;
        if (argc < 3)
        {
```

```
            printf ("contacting %s on port %d\n", serverIp, portno);
        }
        if ((sockfd = socket (AF_INET, SOCK_STREAM, 0)) < 0)
            error (const_cast<char*> ("ERROR opening socket"));
        if ((server = gethostbyname (serverIp) == NULL)
            error (const_cast<char*> ("ERROR, no such host\n"));
        bzero ((char*) &serv_addr, sizeof (serv_addr));
        serv_addr.sin_family = AF_INET;
        bcopy ((char*) server -> h_addr, (char*) &serv_addr.sin_addr.s_addr, server -> h_length);
        serv_addr.sin_port = htons (portno);
        if (connect (sockfd, (struct sockaddr*) &serv_addr, sizeof (serv_addr)) < 0)
            error (const_cast<char*> ("ERROR connecting"));
        for (n = 0; n < 10; n++)
        {
            sendData (sockfd, n);
            data = getData (sockfd);
            printf ("%d -> %d\n", n, data);
        }
        sendData (sockfd, -2);
        close (sockfd);
        return 0;
    }
```

Conclusion:

We have successfully implemented a Client / Server application using Raspberry-Pi.