* **TITLE :** Write a Client/Server application by using Raspberry-pi.
* **PROBLEM STATEMENT :**

Write a server application to be deployed on Raspberry-Pi /Beagle board.

Write client applications to get services from the server application.

* **OBJECTIVES :**
* To understand functionalities of various single board embedded

platforms fundamentals.

* To develop client server application.
* **OUTCOMES :**

The students will be able to :

* Implement an architectural design for IoT for specified requirement.
* CO2: Solve the given societal challenge using IoT
* **SOFTWARE AND HARDWARE REQUIREMENTS :**

Laptop/Computer, Raspberry pi, Ethernet cable, Micro USB

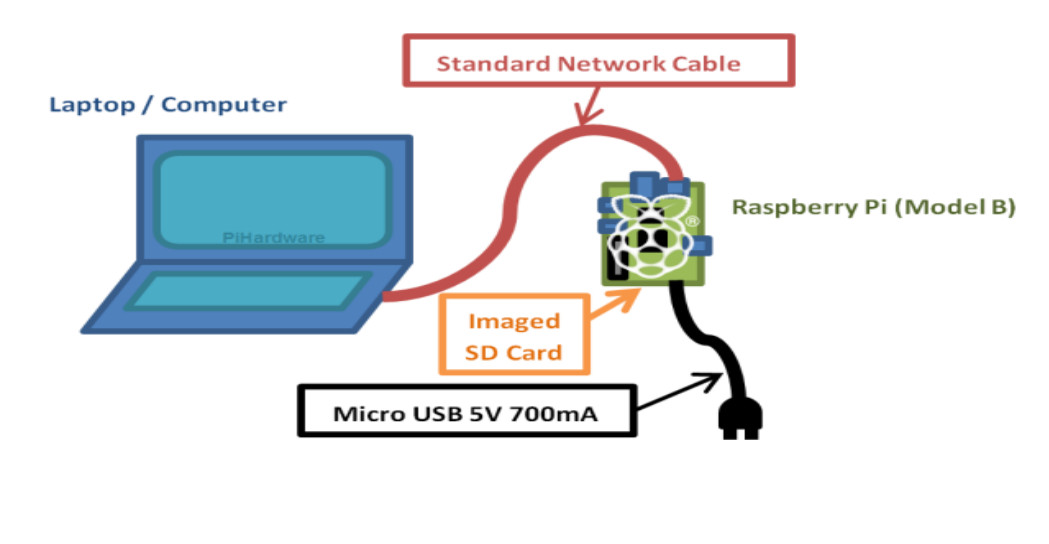
* **THEORY :**

Client–server model is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients.[1] Often clients and servers communicate over a computer network on separate hardware, but both client and server may reside in the same system. A server host runs one or more server programs, which share their resources with clients. A client does not share any of its resources, but it requests content or service from a server. Clients, therefore, initiate communication sessions with servers, which await incoming requests.

Information exchange between a program running on the Raspberry Pi and a partner program running on a remote computer systems becomes important when the Raspberry Pi is a front end of a measurement system and transfers sensor data in real-time to a control station or when a remote PC sends commands a RPi based robot (remote control mode).

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 multiplies 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.

**HARDWARE SETUP :**



* **CODE :**
* **SERVER-SIDE :**

#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);

//--- infinite wait on a connection ---

while ( 1 ) {

printf( "waiting for new client...\n" );

if ( ( newsockfd = accept( sockfd, (struct sockaddr \*) &cli\_addr, (socklen\_t\*) &clilen) ) <

0 )

error( const\_cast<char \*>("ERROR on accept") );

printf( "opened new communication with client\n" );

while ( 1 ) {

//---- wait for a number from client ---

data = getData( newsockfd );

printf( "got %d\n", data );

if ( data < 0 )

break;

data = func( data );

//--- send new data back ---

printf( "sending back %d\n", data );

sendData( newsockfd, data );

}

close( newsockfd );

//--- if -2 sent by client, we can quit ---

if ( data == -2 )

break;

}

return 0;

}

* **CLIENT-SIDE :**

#include *<stdio.h>*

#include *<sys/types.h>*

#include *<sys/socket.h>*

#include *<netinet/in.h>*

#include *<netdb.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) {

*// error( const\_cast<char \*>( "usage myClient2 hostname port\n" ) );*

printf( "contacting %s on port %d\n", serverIp, portno );

*// exit(0);*

}

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 :**

A Client/Server application using Raspberry-pi was successfully implemented.