Read Data from Arduino Using TCP/IP Communication

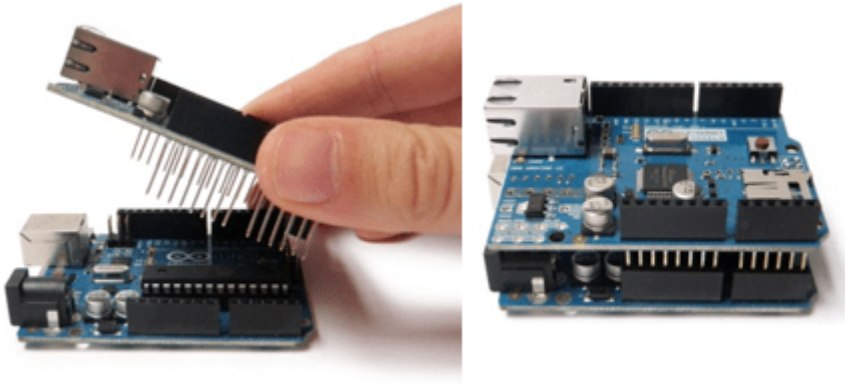
This example shows how to enable callbacks to read sine wave data from an Arduino® Uno using the `tcpserver` interface. The Arduino is configured as a TCP/IP client and connects to the TCP/IP server created in MATLAB® using `tcpserver`.

Copy Command

Connect the Ethernet Shield to Arduino Uno

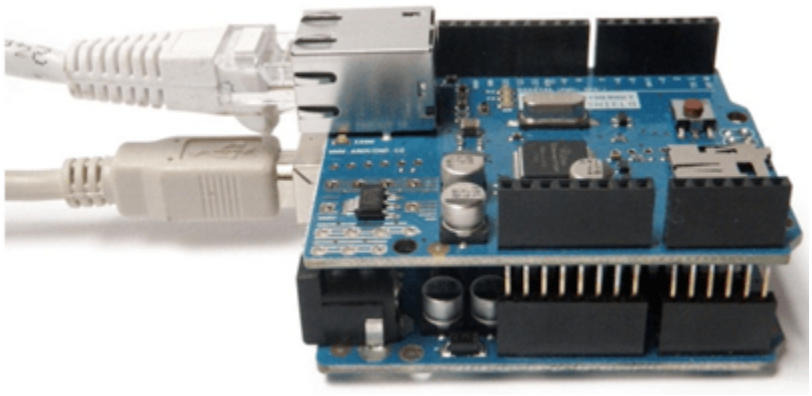
Plug in an Arduino Uno to your computer. Follow these steps to connect the W5100 Ethernet Network Shield to the Arduino Uno and to your network router or a network adapter on the computer.

Place the Ethernet Shield firmly on the Arduino Uno.



Use an RJ45 cable to connect the Arduino Ethernet Shield to one of the following:

- Network router that provides Internet to your computer.
- Network adapter on your computer.



Identify the IP address of the router or network adapter that the Arduino Ethernet Shield is connected to. Specify this IP address in the Arduino program in the **Load Program on the Arduino Uno** section. You also use this IP address as an input argument for `tcpserver` in the **Create the Server** section.

Load Program on the Arduino Uno

Load the following program on the Arduino Uno using the Arduino IDE. This program writes out 250 float values of a sine wave.

```
/*
 TCPIPClient
 Write sine wave data values to the tcpserver object created in MATLAB.
 */

#include <SPI.h>
#include <Ethernet.h>

// Specify the MAC address printed on the Ethernet shield.
// If no MAC address is printed, then use the address shown below.
byte mac[] = {0xDE,0xAD,0xBE,0xEF,0xFE,0xED};

// Specify the server IP address that is used to create the tcpserver object in MATLAB.
// This is the IP address of the router or network adapter that the Arduino Ethernet Shield
// In this example, 192.168.1.81 is the IP address for the server.
IPAddress server(192,168,1,81);

// Set the static IP address for the Arduino Ethernet Shield to act as a TCP/IP client.
// Choose an IP address that is in the same subnet or private network as the server IP addr
// In this example, 192.168.1.177 is the IP address for the Arduino Ethernet Shield. It is
IPAddress ip(192,168,1,177);
IPAddress myDns(192,168,1,1);

// Ethernet client library.
EthernetClient client;

// Command sent by the server.
byte command;

// Sine wave data buffer.
float sineWaveBuffer[250];

// The setup routine runs once when you press reset.
void setup()
{
  // Initialize serial communication.
  Serial.begin(9600);
  while (!Serial)
  {
    ; // Wait for serial port to connect.
  }

  Ethernet.begin(mac,ip,myDns);
  Serial.print("Manually assigned the following IP address to the Arduino:");
  Serial.println();
  Serial.println(Ethernet.localIP());

  // Check for Ethernet hardware.
  if (Ethernet.hardwareStatus() == EthernetNoHardware)
  {
    Serial.println("Ethernet shield was not found.");
```

```
  }

  // Check for Ethernet cable connection.
  if (Ethernet.linkStatus() == LinkOFF)
  {
    Serial.println("Ethernet cable is not connected.");
  }

  Serial.print("Attempting connection to ");
  Serial.print(server);
  Serial.println("...");

  // Attempt to connect to the server running at IP address 192.168.1.81 and port 5000.
  if (client.connect(server,5000))
  {
    Serial.print("Connected to server running at ");
    Serial.println(client.remoteIP());
  }
  else
  {
    Serial.println("Connection to server failed.");
  }

  // Store sine wave data as 250 float values.
  for (int j = 0;j < 250;j++)
  {
    sineWaveBuffer[j] = sin(j*50.0/360.0);
  }
}

// Main processing loop
void loop()
{
  // Block until data is sent by server.
  if (client.available() > 0)
  {
    // Read the command sent by the server.
    command = client.read();

    // Print the command sent by the server.
    Serial.println("The server sent the following command:");
    Serial.println(command);

    if (client.connected() && command == 1)
    {
        // Write sine wave data to the server.
        client.write((const uint8_t *) & sineWaveBuffer, sizeof(sineWaveBuffer));
    }
  }
}
```

Create the Server

Create a `tcpserver` instance using the IP address of the router or network adapter.

In this example, the IP address is `192.168.1.81` and the port number is `5000`. This IP address must be the same one specified in the Arduino program.

```
server = tcpserver("192.168.1.81",5000)
```

```
server =
  TCPServer with properties:

        ServerAddress: "192.168.1.81"
           ServerPort: 5000
            Connected: 0
        ClientAddress: ""
           ClientPort: []
    NumBytesAvailable: 0

  Show all properties, functions
```

Prepare the `tcpserver` Object to Receive Data

Set the `ConnectionChangedFcn` property to `@requestDataCommand`. The callback function `requestDataCommand` is triggered when the Arduino connects to the `tcpserver` object.

```
server.ConnectionChangedFcn = @requestDataCommand;
```

Create a callback function `requestDataCommand` that sends uint8 value of 1 as a command to request the Arduino to send data.

```
function requestDataCommand(src,~)
if src.Connected
    % Display the server object to see that Arduino client has connected to it.
    disp("The Connected and ClientAddress properties of the tcpserver object show that the
    disp(src)

    % Request the Arduino to send data.
    disp("Send the command: 1")
    write(src,1,"uint8");
end
end
```

Set the `BytesAvailableFcnMode` property to "`byte`", the `BytesAvailableFcn` property to `@readArduinoData`, and the `BytesAvailableFcnCount` property to 1000.

```
configureCallback(server,"byte",1000,@readArduinoData);
```

The callback function `readArduinoData` is triggered when 250 sine wave float data points (1000 bytes) are available to be read from the Arduino.

Read Callback Function

Create a callback function `readArduinoData` that reads 250 sine wave data points and plots the result.

```
function readArduinoData(src,~)
% Read the sine wave data sent to the tcpserver object.
src.UserData = read(src,src.BytesAvailableFcnCount/4,'single');

% Plot the data.
plot(src.UserData)
end
```