Windows Phone 8 communicating with Arduino using Bluetooth

This article explains how to communicate with an Arduino board using Windows Phone 8 through Bluetooth.

07 Apr 2013

Introduction

Windows Phone 8 provides new ways to communicate with external hardware which were not available for Windows Phone 7; or it was very hard to implement it. In this post, I'll show you how to communicate with a well know open-source hardware, i.e, Arduino.

As described in Arduino's web site, "Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software". A quick search for "Arduino" on the Internet yields that many cool projects are using it.

As Arduino board is an open-source, there are a lot of sensors and hardware extensions that work with it. So, it is possible to communicate with an Arduino board in many ways such as Ethernet, USB, Bluetooth, etc.

This post focuses on how to accomplish a Bluetooth communication with an Arduino UNO board. This communication will be proved and demonstrated using a small and simple home automation example.

The article also describes the hardware used in this demonstration, how it was configured and the Arduino/Windows Phone 8 code that was implemented to do this communication.

Preparing hardware

Arduino Uno

In this article, I used Arduino Compatible UNO Starter Kit which came with a lot of wires and LEDs.

Bluetooth module

JY-MCU Arduino Bluetooth Wireless Serial Port Module is a Bluetooth module which is used in this article. In spite of its low price, it works fine. But I believe that you can use any Bluetooth module that works with SoftwareSerial.h library.

Proximity sensor

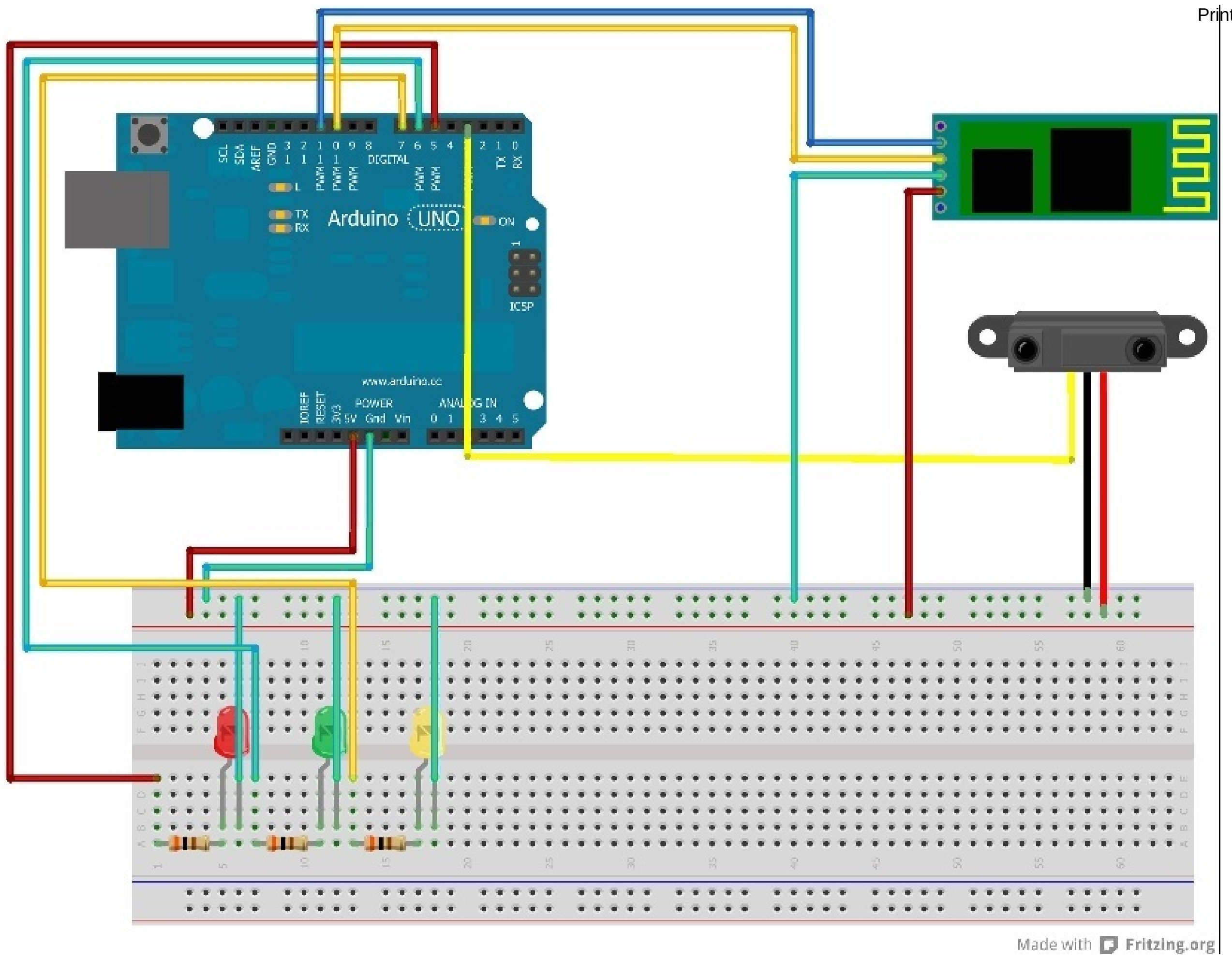
The proximity sensor detects a near body (30cm at maximum). As the Bluetooth communication doesn't depends on this proximity sensor, you can use any other kind of sensor in your project. I just used this one because I already have it. But I could use a temperature sensor, a luminosity sensor, etc. A small change is required in the project should you want to use any of them.

Arduino Infrared Obstacle Avoidance Detection Photoelectric Sensor is used as the proximity sensor in this article.

Configuring Arduino

In order to configure the Arduino development environment on your computer (Windows, Linux or Mac OS X), please visit Getting Started with Arduino.

Once you get all the needed hardware, they must be configured as shown in the following image. In case of any doubt about Arduino, please refer to the Arduino web site (http://www.arduino.cc).

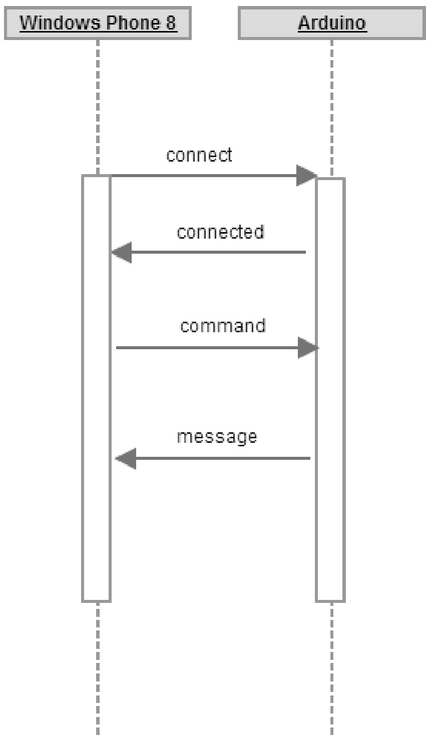


Parts list

- Arduino Uno
- JY-MCU Arduino Bluetooth Wireless Serial Port Module
- Arduino Infrared Obstacle Avoidance Detection Photoelectric Sensor
- 15 wires
- 3 LEDs (Red, Green and Yellow)
- 3 Resistors (300 &Omega)
- 1 Protoboard

Architecture

The communication flow is described below. Since the article focuses on communication part, and to make the things simple and clear, the commands and messages are of type string. If you want to send integer, float or any other type, the code and protocol must be improved.



As shown in the diagram sequence, the Windows Phone 8 and Arduino communicate each other through commands (WP8 --> Arduino) and messages (WP8 <-- Arduino). These commands/messages are of type string (with 255 bytes at maximum).

- Command - Consists of a header of one byte to inform the size of the current command (which limits the command size to 255 characters) and a body that is the command itself.
- Message - Consists of a header of one byte to inform the size of the current message (which limits the message size to 255 characters) and a body that is the message itself.

There are some core codes on both sides; Windows Phone 8 and Arduino; that you can reuse in other projects without changes.

The Windows Phone 8 core code is only the `ConnectionManager` class. This class is used to connect, send commands and receive messages to/from the Arduino. The messages are received through an event handler.

The Arduino core code are the loop() and the sendMessage(char* message) functions, but you must give your own implementation of the sendPeriodicMessages() and processCommand(char* command) functions according to what you want to do. You must also include the "SofwareSerial.h" library in your project and create the related variables and constants to control the serial communication.

Example code

In order to prove that this architecture works fine, one example was implemented. In spite of its small size, this example uses all the features (commands and messages) described in this section.

- The user can turn on/turn off any specific led just on click;
- The user will be notified when someone/something is detected by the proximity sensor on the board;
- After detect someone, the Windows Phone application automatically turn on one specific led (the red one);

In the section Complete example, you can watch this example running or download the source code.

Windows Phone 8 core code

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Diagnostics;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Windows.Networking;
using Windows.Networking.Sockets;
using Windows.Storage.Streams;

namespace BluetoothConnectionManager
{
    /// <summary>
    /// Class to control the bluetooth connection to the Arduino.
    /// </summary>
    public class ConnectionManager
    {
        /// <summary>
        /// Socket used to communicate with Arduino.
        /// </summary>
        private StreamSocket socket;

        /// <summary>
        /// DataWriter used to send commands easily.
        /// </summary>
        private DataWriter dataWriter;

        /// <summary>
        /// DataReader used to receive messages easily.
        /// </summary>
        private DataReader dataReader;

        /// <summary>
        /// Thread used to keep reading data from socket.
        /// </summary>
        private BackgroundWorker dataReadWorker;

        /// <summary>
        /// Delegate used by event handler.
        /// </summary>
        /// <param name="message">The message received.</param>
        public delegate void MessageReceivedHandler(string message);

        /// <summary>
        /// Event fired when a new message is received from Arduino.
        /// </summary>
        public event MessageReceivedHandler MessageReceived;

        /// <summary>
        /// Initialize the manager, should be called in OnNavigatedTo of main page.
        /// </summary>
        public void Initialize()
        {
            socket = new StreamSocket();
            dataReadWorker = new BackgroundWorker();
            dataReadWorker.WorkerSupportsCancellation = true;
            dataReadWorker.DoWork += new DoWorkEventHandler(ReceiveMessages);
        }

        /// <summary>
        /// Finalize the connection manager, should be called in OnNavigatedFrom of main
page.
        /// </summary>
        public void Terminate()
        {
            if (socket != null)
            {
                socket.Dispose();
            }
            if (dataReadWorker != null)
            {
                dataReadWorker.CancelAsync();
            }
        }

        /// <summary>
        /// Connect to the given host device.
        /// </summary>
        /// <param name="deviceHostName">The host device name.</param>
        public async void Connect(HostName deviceHostName)
        {
            if (socket != null)
            {
                await socket.ConnectAsync(deviceHostName, "1");
                dataReader = new DataReader(socket.InputStream);
                dataReadWorker.RunWorkerAsync();
                dataWriter = new DataWriter(socket.OutputStream);
            }
        }

        /// <summary>
        /// Receive messages from the Arduino through bluetooth.
        /// </summary>
        private async void ReceiveMessages(object sender, DoWorkEventArgs e)
        {
            try
            {
                while (true)
                {
                    // Read first byte (length of the subsequent message, 255 or less).
                    uint sizeFieldCount = await dataReader.LoadAsync(1);
                    if (sizeFieldCount != 1)
                    {
                        // The underlying socket was closed before we were able to read
the whole data.
                        return;
                    }

                    // Read the message.
                    uint messageLength = dataReader.ReadByte();
                    uint actualMessageLength = await
dataReader.LoadAsync(messageLength);
                    if (messageLength != actualMessageLength)
                    {
                        // The underlying socket was closed before we were able to read
the whole data.
                        return;
                    }
                    // Read the message and process it.
                    string message = dataReader.ReadString(actualMessageLength);
                    MessageReceived(message);
                }
            }
            catch (Exception ex)
            {
                Debug.WriteLine(ex.Message);
            }

        }

        /// <summary>
        /// Send command to the Arduino through bluetooth.
        /// </summary>
        /// <param name="command">The sent command.</param>
        /// <returns>The number of bytes sent</returns>
        public async Task<uint> SendCommand(string command)
        {
            uint sentCommandSize = 0;
            if (dataWriter != null)
            {
                uint commandSize = dataWriter.MeasureString(command);
                dataWriter.WriteByte((byte)commandSize);
                sentCommandSize = dataWriter.WriteString(command);
                await dataWriter.StoreAsync();
            }
            return sentCommandSize;
        }
    }
}
```

Arduino core code

```
#include <SoftwareSerial.h>

const int TX_BT = 10;
const int RX_BT = 11;

SoftwareSerial btSerial(TX_BT, RX_BT);

//Frequency to send periodic messages to Windows Phone, in milliseconds.
//Core code.
const unsigned long periodicMessageFrequency = 5000;
unsigned long time = 0;

//Process the incoming command from Windows Phone.
//It should be changed according to what you want to do.
void processCommand(char* command) {
}

//Send a message back to the Windows Phone.
//Is can't be changed.
void sendMessage(char* message) {
  int messageLen = strlen(message);
  if(messageLen < 256) {
    btSerial.write(messageLen);
    btSerial.print(message);
  }
}

//Send a set of periodic messages to the Windows Phone.
//It should be changed according to what you want to do.
//This message could be a sensor data, like a thermometer data.
void sendPeriodicMessages() {
}

//Setup Arduino function
void setup() {
  Serial.begin(9600);
  Serial.println("USB Connected");
  btSerial.begin(9600);
}

//Loop Arduino function
//It can't be changed
void loop() {
  if(btSerial.available()) {
      int commandSize = (int)btSerial.read();
      char command[commandSize];
      int commandPos = 0;
      while(commandPos < commandSize) {
        if(btSerial.available()) {
          command[commandPos] = (char)btSerial.read();
          commandPos++;
        }
      }
      command[commandPos] = 0;
      processCommand(command);
  }
  unsigned long currentTime = millis();
  if((currentTime - time) > periodicMessageFrequency) {
    sendPeriodicMessages();
    time = currentTime;
  }
}
```

Complete example

If you already have the Arduino board configured as described in the previous section, you can download the complete example and test it.

Source list

- The Windows Phone 8 code: (here)
- Arduino code: (here)

If you just want to see the example running, watch the below video.

More possibilities

Talking to Arduino, using Speech API.

Arduino talking to user, using text-to-speech feature.

Temperature control.

Thanks

A special thanks to xharada ! He gave me the idea of try to do this.