Windows Phone 8 comunicando com Arduino usando Bluetooth

Este artigo explica como implementar a comunicação de uma aplicação Windows Phone 8 com uma placa Arduino usando Bluetooth.

Introdução

A plataforma Windows Phone 8 provê novas formas de comunicação com dispositivos externos, formas que não estavam disponíveis, ou eram muito difíceis de implementar, na plataforma Windows Phone 7. Neste artigo, irei mostrar como comunicar com o conhecido hardware open-source Arduino.

Como está descrito no site do projeto Arduino, "Arduino é uma plataforma de prototipação eletrônica open-source, baseada em hardware e software flexíveis e fáceis de usar". Uma busca rápida por "Arduino" na Internet retorna muito projetos interessantes que são feitos com Arduino.

Como Arduino é open-source, existe um conjunto grande de sensores e extensões de hardware que funcionam com ele. Logo, é possível comunicar com uma placa Arduino de muitas formas, como Ethernet, USB, Bluetooth, etc.

Este artigo trata somente da comunicação via Bluetooth com uma placa Arduino UNO. Esta comunicação vai ser demonstrada através de um pequeno exemplo de automação residencial.

O artigo também descreve o hardware usado na demonstração, como este foi configurado e o código Arduino/Windows Phone 8 que foi implementado para realizar esta comunicação.

Preparando o hardware

Arduino Uno

Neste artigo, Eu utilizei este kit Arduino que já vem com muitos fios, leds e sensores.

Módulo Bluetooth

JY-MCU é o módulo Bluetooth utilizado neste artigo. Apesar do preço baixo, ele funciona muito bem. Porém, acredito que qualquer outro módulo Bluetooth pode ser utilizado para este exemplo.

Sensor de proximidade

O sensor de proximidade detecta corpos a uma distância de no máximo 30 cm. Como a comunicação Bluetooth não depende deste sensor, você pode utlizar qualquer outro tipo de sensor. Eu utilizei este somente porquê era o que eu já tinha em mãos. Mas poderia ter utlizado um sensor de temperatura, luminosidade, etc. Contudo, uma pequena mudança no projeto é requerida se o sensor for alterado.

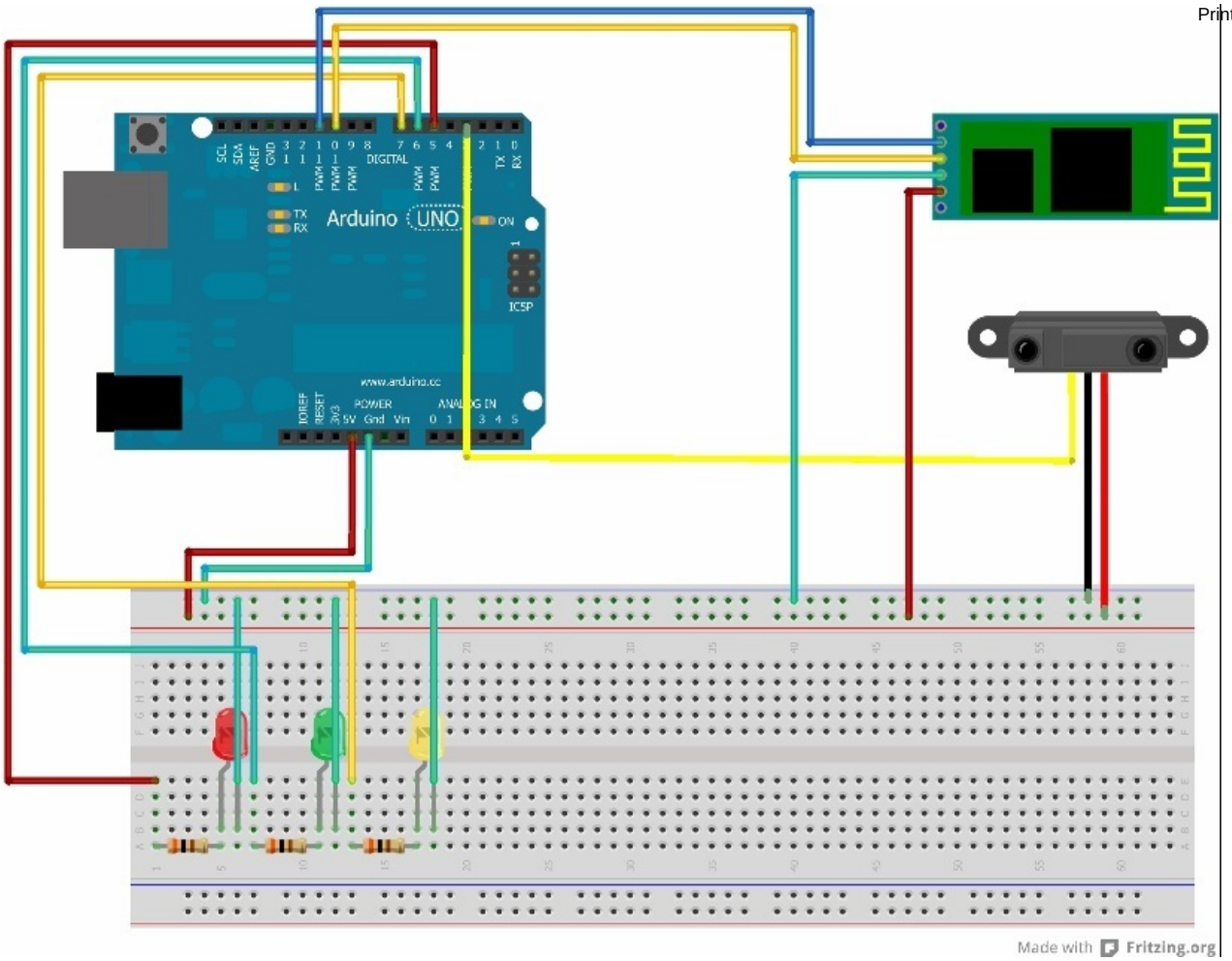
O sensor de proximidade utlizado neste projeto foi o Arduino Infrared Obstacle Avoidance Detection Photoelectric Sensor.

Configurando o Arduino

Para configurar o ambiente de desenvolvimento Arduino em seu computador (Windows, Linux or Mac OS X), visite Getting Started with Arduino.

Uma vez que você tenha todo o hardware em mãos, eles devem ser configurados como mostra a figura abaixo. Em caso de dúvidas sobre como manusear o Arduino, visite o site (http://www.arduino.cc).

(C) Copyright Nokia 2014. All rights reserved.



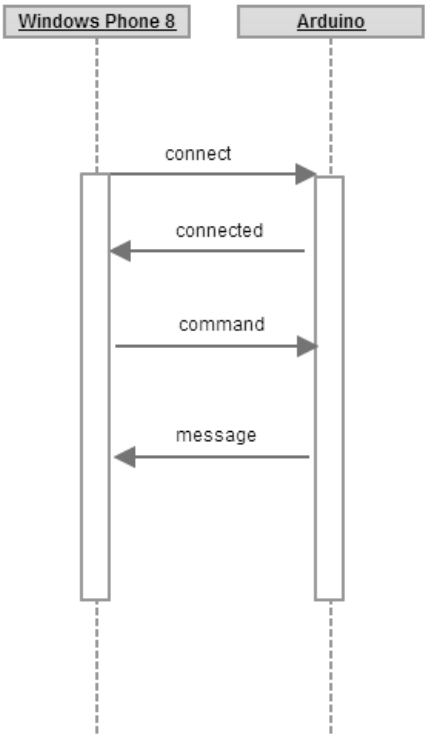
Lista de partes

- Arduino Uno
- JY-MCU Arduino Bluetooth Wireless Serial Port Module
- Arduino Infrared Obstacle Avoidance Detection Photoelectric Sensor
- 15 fios
- 3 LEDs (Vermelho, verde e amarelo)
- 3 Resistores (300 &Omega)
- 1 Protoboard

Arquitetura

O fluxo de comunicação é descrito abaixo. Como o artigo tem foco na comunicação, e para manter as coisas simples, os comandos e mensagens recebidas são sempre cadeias de caracteres. Se você deseja enviar outros tipos de dados, como inteiro, ponto flutuante ou outro tipo, o código e o protocolo devem ser atualizados.

(C) Copyright Nokia 2014. All rights reserved.



Como mostrado no diagrama de sequência, o Windows Phone 8 e o Arduino se comunicam através de comandos (WP8 --> Arduino) e mensagens (WP8 <-- Arduino). Estes comandos/mensagens são do tipo texto (com 255 bytes no máximo).

- Comando - Um cabeçalho de um byte, contedo o tamanho do comando (o qual é limitado em 255 caracteres) e o comando em si.
- Mensagens - Um cabeçalho de um byte, contedo o tamanho da mensagem (o qual é limitado em 255 caracteres) e a mensagem em si.

Existem funções/classes prontas em ambos os lados, Windows Phone 8 e Arduino, que você pode reutilizar em outros projetos sem mudanças.

No Windows Phone 8 temos somente a classe `ConnectionManager`. Esta classe é usada para conectar, enviar comandos e receber mensagens do/para o Arduino. As mensagens são recebidas através de um event handler.

No Arduino temos o conteúdo da função loop() e a função `sendMessage(char* message)`, porém você deve fornecer sua própria implementação para as funções `sendPeriodicMessages()` e `processCommand(char* command)` de acordo com o que você deseja fazer. Você também deve incluir a biblioteca `SoftwareSerial.h` no seu código Arduino e criar as varíaveis e constantes relacionadas à comunicação serial.

Código de exemplo

Para provar que esta arquitetura funcionar, um exemplo foi implementado. Apesar da simplicidade, este exemplo utiliza todas as características apresentadas (comandos e mensages).

- O usuário pode ligar e desligar um LED específico.
- O usuário será notificado quando alguém ou alguma coisa for detectado pelo sensor de proximidade.
- Após detectar algupem, a aplicação Windows Phone irá automaticamente ligar o LED vermelho.

Na seção Complete example, você pode assitir este exemplo em execução ou baixar o código fonte.

Código do Windows Phone 8

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Diagnostics;
using System.Linq;
using System.Text;
```


(C) Copyright Nokia 2014. All rights reserved.

```
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.
```


(C) Copyright Nokia 2014. All rights reserved.

```
        /// </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;
                    }
```


(C) Copyright Nokia 2014. All rights reserved.

```
                    // 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;
        }
    }
}
```

Código do Arduino

```
#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);
```


(C) Copyright Nokia 2014. All rights reserved.

```
  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;
  }
}
```

Exemplo completo

Se você já tem a placa Arduino configurada como descrito na seção anterior, você pode baixar o código complete e testá-lo.

Lista de fontes:

- Código Windows Phone 8: (File:BluetoothClientWP8.zip)
- Código Arduino: (File:BluetoothArduino.zip)

Se você deseja somente ver o exemplo em execução, assista o vídeo abaixo.

(C) Copyright Nokia 2014. All rights reserved.

Mais possibilidades

Falar com o Arduino, usando a Speech API.

Arduino falando com o usuário, usando o TTS API.

Controle de temperatura.

Obrigado

Um obrigado para xharada , que me deu a idéia de tentar fazer isto.

Version Hint

Windows Phone: [[Category:Windows Phone]]
[[Category:Windows Phone 7.5]]
[[Category:Windows Phone 8]]

Nokia Asha: [[Category:Nokia Asha]]
[[Category:Nokia Asha Platform 1.0]]

Series 40: [[Category:Series 40]]
[[Category:Series 40 1st Edition]] [[Category:Series 40 2nd Edition]]
[[Category:Series 40 3rd Edition (initial release)]] [[Category:Series 40 3rd Edition FP1]] [[Category:Series 40 3rd Edition FP2]]
[[Category:Series 40 5th Edition (initial release)]] [[Category:Series 40 5th Edition FP1]]
[[Category:Series 40 6th Edition (initial release)]] [[Category:Series 40 6th Edition FP1]] [[Category:Series 40 Developer Platform 1.0]] [[Category:Series 40 Developer Platform 1.1]] [[Category:Series 40 Developer Platform 2.0]]

Symbian: [[Category:Symbian]]
[[Category:S60 1st Edition]] [[Category:S60 2nd Edition (initial release)]] [[Category:S60 2nd Edition FP1]] [[Category:S60 2nd Edition FP2]] [[Category:S60 2nd Edition FP3]]
[[Category:S60 3rd Edition (initial release)]] [[Category:S60 3rd Edition FP1]] [[Category:S60 3rd Edition FP2]]
[[Category:S60 5th Edition]]
[[Category:Symbian^3]] [[Category:Symbian Anna]] [[Category:Nokia Belle]]

 (C) Copyright Nokia 2014. All rights reserved.

(C) Copyright Nokia 2014. All rights reserved.