Simple home automation project with a Windows Phone 8 an Arduino and a TV

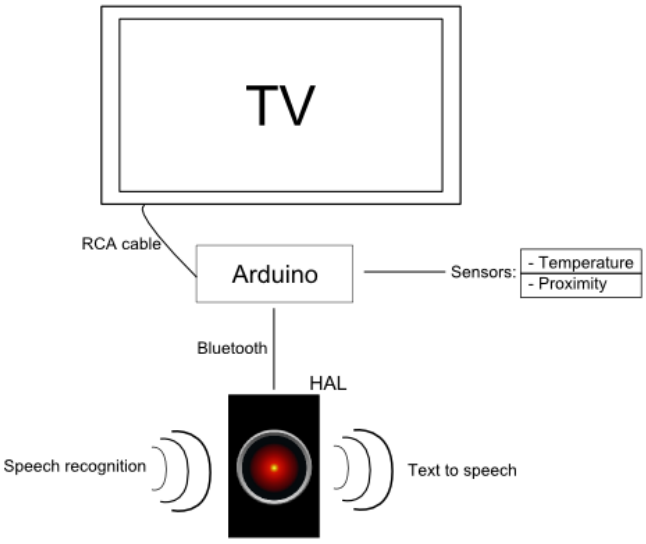
This article explains how to do a simple home automation project with a Windows Phone 8, an Arduino and a TV.

Introduction

In a previous article Windows Phone 8 communicating with Arduino using Bluetooth, I've shown how to implement a Windows Phone 8 application that communicates with an Arduino board using Bluetooth.

In this post I'll show you how to improve the project created previously to develop a simple home automation project that is able to communicate with an Arduino board to get data from external sensors, show some information on TV and listen/talk to the user.

The example application, named HAL (I hope you've ever seen 2001), is able to receive voice commands from user, transcript them and send these commands to the Arduino (as shown in Windows Phone 8 communicating with Arduino using Bluetooth). These commands can be used to draw basic shapes and write text on TV; as well as to get sensor data from any sensor connected to the Arduino. The example's architecture is shown below.



In fact, this post is more like an extension of Windows Phone 8 communicating with Arduino using Bluetooth and its main purpose is to show more things that you can do with Windows Phone 8 and Arduino. If you've already implemented the Bluetooth communication between Windows Phone 8 and Arduino shown in the previous post, the procedure described here will be quite easy; and this is a good thing! It'll prove that developing simple (but cool) Home Automation projects aren't rocket science or expensive projects nowadays.

In the next sections, you will learn the architecture, schematic and code of the project; as well as any external library or code used in this project. A special thanks to the Arduino TV Out project, it made the communication between Arduino and TV really simple. I would also like to thank rudyhuyn for the Speech recognition using a SGRS file post which helped me in implementing the speech recognition.

Architecture

The project's architecture can be split between two parts:

1. The Windows Phone 8 application: It is able to receive voice commands from user, transcript them into known commands and send them to the Arduino board through Bluetooth. It is also able to receive messages from the Arduino board and show/talk them to the user.
2. The Arduino board application: The main responsibility of the Arduino board is to receive commands from the Windows Phone 8 application as well as send messages to the Windows Phone 8 application through Bluetooth. It is also responsibility of the Arduino board to show data on TV (through a RCA cable) and to receive data from any connected sensor. In this example, the Arduino board is connected to a temperature sensor and a proximity sensor.

Current features list:

- Request the current temperature value
- Detect movement using proximity sensor
- Play a Tic Tac Toe game on TV

Windows Phone 8 project

(C) Copyright Nokia 2014. All rights reserved.

The main class of the project is the `Hal9000` class which has three classes that compound the project: `ArduinoManager`, `SpeechRecognizerStateManager` and `SpeechSynthesizer`. Each one will be described as follows:

1. `ArduinoManager`:

   The `ArduinoManager` class is a wrapper to commands acceptable by the Arduino. It has methods to draw basic shapes and text on TV and also a method to retrieve the current temperature. It also receives a message from Arduino when a person is detected by the proximity sensor. In order to access the Arduino board, it has an instance of `ConnectionManager` class that is very similar to the `ConnectionManager` class from this post. But now, this class is able to send commands as byte array. It make possible to send smaller commands to the Arduino.

2. `SpeechSynthesizer`:

   The `Synthesizer` is only used to speak to the user.

3. `SpeechRecognizerStateManager`:

   It's the `SpeechRecognizerStateManager` shown in this post; This class manages the available states and send the speech writing to the active state. This application has three available states: 'Play', 'HomeAutomation' and 'Initial'. If the user speaks "row 1 column 2" while in 'Initial' state, nothing happens; But speaking the same in the 'Play' state will update the game state and send correct draw commands to the Arduino, in order to update the TV screen. The user can also change from a state to another, for example speaking 'play' while in 'Initial' or 'HomeAutomation' states; There states are classes in the project (`PlayState`, `HomeAutomationState` and `InitialState`) and each one of them is subclass of `SpeechRecognizerState` class.

Windows Phone 8 Code

Below is shown the Hal state base class. The 'Play', 'Initial' and 'HomeAutomation' states are subclasses of this class. After create the state instances, you just have to put them in the `SpeechRecognizerStateManager` instance and calll the `SpeechRecognizerStateManager.Initialize(string firstState)` method.

```
/// <summary>
/// Base HAL state. It inherits from SpeechRecognizerState, so it is able to receive
speech writing commands.
/// </summary>
public abstract class HalState : SpeechRecognizerState
{
 /// <summary>
 /// The HAL9000 instance.
 /// </summary>
 public Hal9000 Hal
 {
  private set;
  get;
 }

 /// <summary>
 /// Create a Hal State instance, with the given parameters.
 /// </summary>
 /// <param name="hal">The HAL9000 instance</param>
 /// <param name="stateKey">The stateKey</param>
 /// <param name="grammarUri">The grammar Uri</param>
 public HalState(Hal9000 hal, string stateKey, Uri grammarUri) : base(stateKey,
grammarUri)
 {
  Hal = hal;
 }

 /// <summary>
 /// Handle the received commands from Arduino.
 /// </summary>
 /// <param name="message">The received message.</param>
 public abstract void HandleReceivedMessage(string message);

}
```


(C) Copyright Nokia 2014. All rights reserved.

The `HomeAutomationState` class inherits from `HalState` class and control the voice commands while in home automation state.

```
/// <summary>
/// Class to control the home automation state.
/// </summary>
public class HomeAutomationState : HalState
{
 /// <summary>
 ///  The HomeAutomationState constructor. It receives the Hal9000 instance and set the
state key to "home" and the acceptable grammar source to "HomeAutomationGrammar.xml".
 /// </summary>
 public HomeAutomationState(Hal9000 hal)
  : base(hal, "home", new Uri("ms-appx:///HomeAutomationGrammar.xml", UriKind.Absolute))
 {
 }

 /// <summary>
 /// Process the retrieved speech writing.
 /// </summary>
 /// <param name="speechWriting"></param>
 public override void Process(string speechWriting)
 {
  if (speechWriting.Equals("temperature"))
  {
   //Get temperature from Arduino.
   Hal.ArduinoManager.GetTemperature();
  }
  else if (speechWriting.Equals("play"))
  {
   //Change to Play state.
   MoveTo("play");
  }
 }

 /// <summary>
 /// Called after entering the Home Automation state.
 /// </summary>
 public override async void Enter()
 {
  await Hal.Synthesizer.SpeakTextAsync("What do you want?");
  Hal.ArduinoManager.ClearScreen();
  Hal.ArduinoManager.PrintText(10, 10, "Temperature");
 }

 public override void Exit()
 {
  //Do nothing.
 }

 /// <summary>
 /// Handle the received message from Arduino.
 /// </summary>
 /// <param name="message">The message.</param>
 public override async void HandleReceivedMessage(string message)
 {
  if (message.Split(':')[0].Equals("temperature"))
  {
   Hal.ArduinoManager.ClearScreen();
   Hal.ArduinoManager.PrintText(2, 2, "Temperature is");
   Hal.ArduinoManager.PrintText(2, 20, message.Split(':')[1]);
   await Hal.Synthesizer.SpeakTextAsync(message.Split(':')[1]);
```


(C) Copyright Nokia 2014. All rights reserved.

```
  }
  else
  {
   await Hal.Synthesizer.SpeakTextAsync(message);
  }
 }
}
```

The `Hal9000` is the main class of the project. It has access to the `ArduinoManager`, `SpeechRecognizerStateManager` and `SpeechSynthesizer` classes.

```
    /// <summary>
    /// The Hal9000 class.
    /// </summary>
    public class Hal9000
    {
        /// <summary>
        /// The arduino manager.
        /// </summary>
        public ArduinoManager ArduinoManager;

        /// <summary>
        /// The speech recognizer state manager.
        /// </summary>
        public SpeechRecognizerStateManager SpeechManager;

        /// <summary>
        /// The speech synthesizer.
        /// </summary>
        public SpeechSynthesizer Synthesizer;

        public Hal9000()
        {
            // Create the manager instances.
            ArduinoManager = new ArduinoManager(new ConnectionManager());
            SpeechManager = new SpeechRecognizerStateManager();
            Synthesizer = new SpeechSynthesizer();
            // Creating the available states
            // After put them in the SpeechRecognizerStateManager, you just have to call
the SpeechManager.Initialize("initial") method.
            SpeechManager.AddState(new HomeAutomationState(this));
            SpeechManager.AddState(new PlayState(this));
            SpeechManager.AddState(new InitialState(this));

            // Query for a voice that speaks English.
            IEnumerable<VoiceInformation> halVoices = from voice in InstalledVoices.All
                                                      where voice.Language == "en-US"
&& voice.Gender == VoiceGender.Male
                                                      select voice;
            if (halVoices.Count() > 0)
            {
                // Set the voice as identified by the query.
                Synthesizer.SetVoice(halVoices.ElementAt(0));
            }
        }

        /// <summary>
        /// Initialize the HAL 9000.
        /// </summary>
        internal void Initialize()
        {
            ArduinoManager.ConnectionManager.MessageReceived +=
```

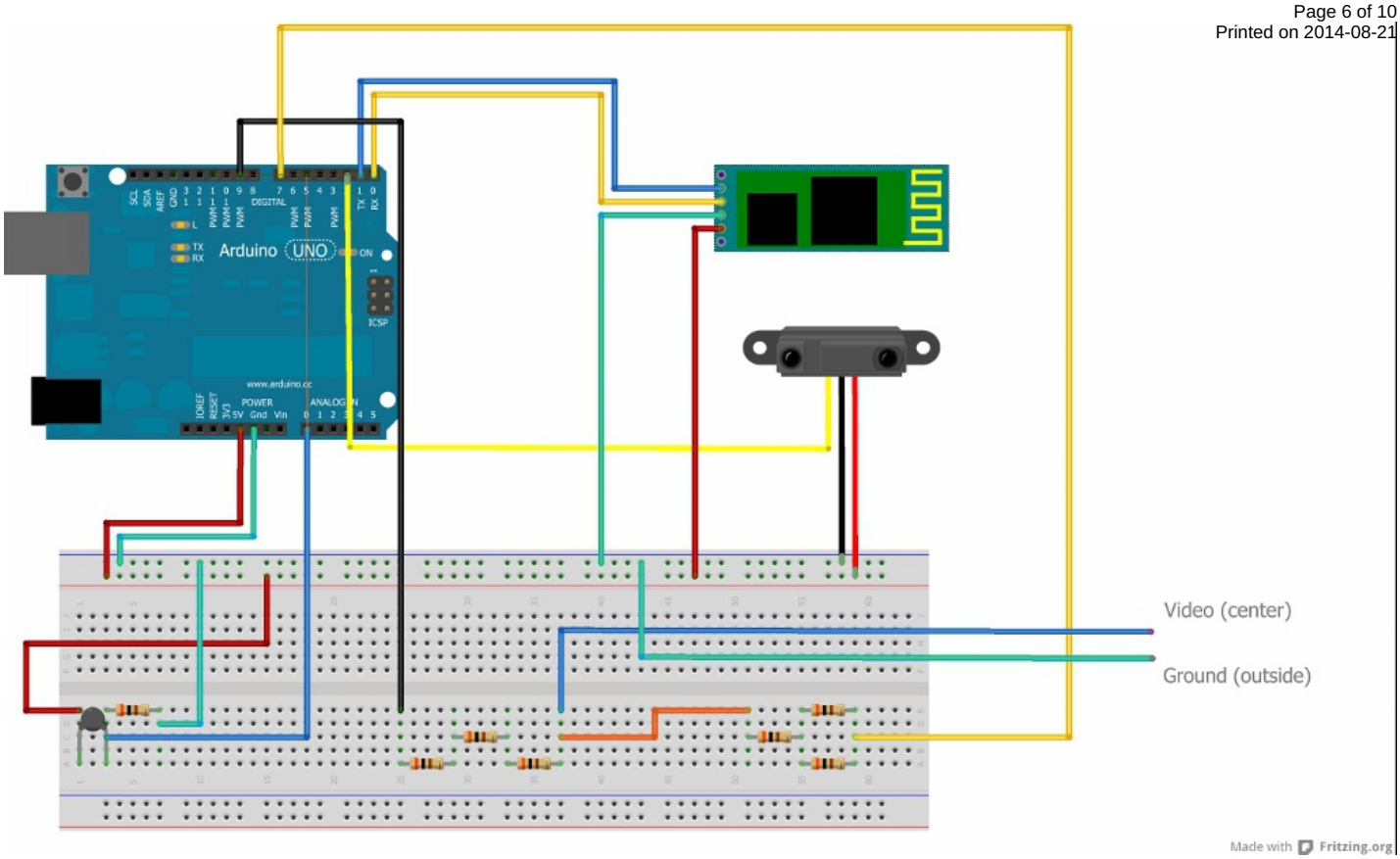

(C) Copyright Nokia 2014. All rights reserved.

```
btConnectionManager_MessageReceived;
            ArduinoManager.ConnectionManager.Initialize();
            // Initialize the speech recognizer manager in the 'initial' state.
            SpeechManager.Initialize("initial");
        }

        /// <summary>
        /// Terminate the HAL 9000.
        /// </summary>
        internal void Terminate()
        {
            ArduinoManager.ConnectionManager.Terminate();
            SpeechManager.Terminate();
        }

        /// <summary>
        /// Receive any message from Arduino and send it to the current state.
        /// </summary>
        /// <param name="message"></param>
        private void btConnectionManager_MessageReceived(string message)
        {
            ((HalState)SpeechManager.CurrentState).HandleReceivedMessage(message);
        }

        /// <summary>
        /// Connect to the Arduino. The Arduino name must be the default name 'linvor'.
        /// </summary>
        /// <returns></returns>
        public async Task<bool> SearchAndConnect()
        {
            PeerFinder.AlternateIdentities["Bluetooth:Paired"] = "";
            var pairedDevices = await PeerFinder.FindAllPeersAsync();

            if (pairedDevices.Count == 0)
            {
                Debug.WriteLine("No paired devices were found.");
            }
            else
            {
                foreach (var pairedDevice in pairedDevices)
                {
                    if (pairedDevice.DisplayName.Equals("linvor"))
                    {
                        ArduinoManager.ConnectionManager.Connect(pairedDevice.HostName);
                        return true;
                    }
                }
            }
            return false;
        }
    }
```

Arduino project

The Arduino schematic is shown below:

(C) Copyright Nokia 2014. All rights reserved.



This is the parts list to build this schematic:

- 1 Arduino Uno
- 1 JY-MCU Arduino Bluetooth Wireless Serial Port Module
- 1 Arduino Infrared Obstacle Avoidance Detection Photoelectric Sensor
- 1 NTC - Negative temperature coefficient
- 1 RCA cable
- 13 wires
- 7 Resistors (300 &Omega)
- 1 Protoboard

Arduino Code

The Arduino code is very similar to the code shown in the previous post. To install the TV out library in your Arduino development environment, take a look at this site.

```
// Include TVout
#include <TVout.h>
#include <fontALL.h>

TVout TV;

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

//The same commands declared in Windows Phone 8. Each command has a fixed number of
arguments.
const byte DRAW_RECT_CMD = 0x01;
const byte DRAW_CIRCLE_CMD = 0x02;
const byte DRAW_LINE_CMD = 0x03;
const byte CLEAR_CMD = 0x04;
const byte PRINT_CMD = 0x05;
const byte GET_TEMP_CMD = 0x06;

const char* INVALID_PARAMETERS_MSG = "Invalid Parameters";
```

 (C) Copyright Nokia 2014. All rights reserved.

```
const char* DRAW_RECT_OK = "Draw rect ok";
const char* DRAW_CIRCLE_OK = "Draw circle ok";
const char* DRAW_LINE_OK = "Draw line ok";
const char* CLEAR_OK = "Clear ok";
const char* PRINT_OK = "Print ok";

const byte PIN_TEMPERATURE = 0;

const int WIDTH = 104;
const int HEIGHT = 96;

//Process the incoming command from Windows Phone.
void processCommand(byte commandSize, byte* command) {
  if(command[0] == CLEAR_CMD) {
    TV.clear_screen();
  } else if(command[0] == DRAW_RECT_CMD) {
    if((command[1] + command[3] > WIDTH) || (command[2] + command[4] > HEIGHT) ||
command[1] < 0 || command[2] < 0) {
      sendMessage(INVALID_PARAMETERS_MSG);
      return;
    }
    TV.draw_rect(command[1], command[2], command[3], command[4], command[5]);
  } else if(command[0] == DRAW_CIRCLE_CMD) {
    if((command[1] + command[3] > WIDTH) || (command[2] + command[3] > HEIGHT) ||
(command[1] - command[3] < 0) || (command[2] - command[3] < 0)) {
      sendMessage(INVALID_PARAMETERS_MSG);
      return;
    }
    TV.draw_circle(command[1], command[2], command[3], command[4]);
  } else if(command[0] == DRAW_LINE_CMD) {
    if(command[1] > WIDTH || command[1] < 0 || command[3] > WIDTH || command[3] < 0 ||
command[2] > HEIGHT || command[2] < 0 || command[4] > HEIGHT || command[4] < 0) {
      sendMessage(INVALID_PARAMETERS_MSG);
      return;
    }
    TV.draw_line(command[1], command[2], command[3], command[4], command[5]);
  } else if(command[0] == PRINT_CMD) {
    if(commandSize < 4) {
      sendMessage(INVALID_PARAMETERS_MSG);
      return;
    }
    char text[commandSize - 3];
    for(int i = 0;i < sizeof(text);i++) {
      text[i] = (char)command[i + 3];
    }
    TV.print(command[1], command[2], text);
  } else if(command[0] == GET_TEMP_CMD) {
    int valTemp = analogRead(PIN_TEMPERATURE);
    char tempAsStr[30];
    sprintf(tempAsStr, "temperature:%d", valTemp);
    sendMessage(tempAsStr);
  }
}

//Send a message back to the Windows Phone.
void sendMessage(const char* message) {
  int messageLen = strlen(message);
  if(messageLen < 256) {
    Serial.write(messageLen);
    Serial.print(message);
  }
}
```


(C) Copyright Nokia 2014. All rights reserved.

```
//Send a set of periodic messages to the Windows Phone.
//This message could be a sensor data, like a thermometer data.
void sendPeriodicMessages() {
}

//Setup Arduino function
void setup() {
  //Configure bluetooth
  Serial.begin(9600);
  //Configure tv out
  int beginResult = TV.begin(PAL, WIDTH, HEIGHT);
  TV.select_font(font6x8);
  TV.clear_screen();
  TV.println(6, 5, "Waiting...");

  attachInterrupt(0, moveDetected ,FALLING);
}

void moveDetected() {
  sendMessage("Move detected");
}

//Loop Arduino function
void loop() {
  if(Serial.available()) {
      byte commandSize = Serial.read();
      byte* command = (byte*)malloc(commandSize * sizeof(byte));
      int commandPos = 0;
      while(commandPos < commandSize) {
        if(Serial.available()) {
          command[commandPos] = Serial.read();
          commandPos++;
        }
      }
      command[commandPos] = 0;
      processCommand(commandSize, command);
      free(command);
  }
  unsigned long currentTime = millis();
  if((currentTime - time) > periodicMessageFrequency) {
    sendPeriodicMessages();
    time = currentTime;
  }
}
```

Example Running

Playing and requesting temperature value.

Warning: You will see that the application goes to the 'Play' state by itself. It's my fault, i should had used larger strings in the grammar file. Using a small string like 'play' makes any external noise to be considered as a valid grammar. I also could try to put some mark command in the grammar; Something like 'Do it Hal'

 (C) Copyright Nokia 2014. All rights reserved.

Move detected example

Source code:

- File:Wp8 arduino tv project source.zip
- File:Tvout bluetooth.zip

Warning: This example only works if the device is configured to 'English - United States' location. Any other location will break the application on start-up.

References

Projects and posts that I used to implement this idea.

- Arduino TV out project
- Speech recognition using a SGRS file
- Working with grammars on Windows Phone 8
- Managing loaded grammars to optimize recognition for Windows Phone 8
- A state machine to handle speech recognition feature

Conclusion

This is just an extension of my previous post about Windows Phone 8/Arduino to show how it can be easily improved. Now, I hope that other projects using Arduino or another external hardware will be shown soon.

The communication between a Series 40 device and an Arduino board is also possible, as shown in this post Series 40 communicating with Arduino using Bluetooth. The Series 40 is cheaper than Lumia devices and a lot of them are sold in poor countries; Cheap home automation projects with Arduino that runs on Series 40 devices is accessible and can be very usefull.

 (C) Copyright Nokia 2014. All rights reserved.

 (C) Copyright Nokia 2014. All rights reserved.