ISSN No. 0976-5697

Volume 6, No. 6, June - July 2015

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

An Inexpensive Wearable Android Accessory

Kaustav Basu
Department of Computer Science
St. Xavier's CollegeKolkata, India

Prabal Banerjee
Chennai Mathematical Institute
Chennai, India

Proyag Pal
Department of Computer Science
St. Xavier's CollegeKolkata, India

Abstract: The project aims to design and implement a prototype for a peripheral device built to function with any Android device. The device behaves as a notification display for an Android device (a phone or tablet) on a wristwatch-like accessory, and displays an ordinary digital watch when there are no notifications to be displayed. The aim is not to develop an altogether new technology, but to create a simpler, more affordable and accessible alternative to similar existing devices. The application of non-proprietary open-source technologies like Android and Arduino leaves the device open to modifications as per the requirements of users, while simultaneously reducing the cost drastically. Android's overwhelming popularity ensures the accessory is available to a huge number of devices. Hence, from this choice of technologies, it is clear that our objective is to use freely available open-source software and technologies to make this device universally compatible to Android devices.

Keywords:–Arduino, Android, Wearable, open source, phone accessory, Amarino, Bluetooth, smart watch.

I. INTRODUCTION

In the modern world, smart phones are dominating the mobile market. The Android[1] Operating System, developed by Google[2] , accounts for a vast majority of these devices.

Recently, technology giants have developed wearable accessories for their smart phones. These accessories have a myriad of features. But the basic function of these devices is to show the time and notify the user when the phone receives any notifications, such as a call or a text message. However, from the common man's perspective, these devices are very expensive. In fact, it costs more than an average high-end smart phone. A number of manufacturers are also creating such devices. But the major drawback of most of these devices is that despite their prices and numerous features, each of these accessories are only compatible with the smart phone they are sold with. The Android Wear project, which has been initiated by Google, is still in its development phase. The wearable peripheral aims to work with most Android devices. All these initiatives taken by several companies clearly indicate that there is a high demand for such wearables in the smart-device world. But presently such devices are accessible to a very limited number of users.

So we have decided to create an accessory which will be accessible to a wider array of consumers by using open source[3] technologies. This will result in wider compatibility and lower prices.

II. CHOICE OF TECHNOLOGIES

A. *Why Android?*

The major mobile operating systems are Android, iOS, Windows and Symbian. Android accounts for nearly 80% of the mobile OS market[4]. It is developer friendly and has many support modules for Arduino. So it was easier for us to implement the prototype based on Android. We also had previous experience in Android programming, so we felt comfortable choosing this platform.

Android's source code is released by Google under the Apache License[5]; this permissive licensing allows the software to be freely modified and distributed by device manufacturers, wireless carriers and enthusiast developers. Most Android devices ship with a combination of open source and proprietary software. As of October 2013, Android has the largest number of applications ("apps"), available for download in Google Play store which has had over 1 million apps published, and over 50 billion downloads. A developer survey conducted in April–May 2013[6] found that Android is the most used platform among developers: it is used by 71% of the mobile developer population.

B. *Why Arduino?*

Arduino[7] is a single-board microcontroller, intended to make the application of interactive objects or environments more accessible. The hardware consists of an open-source hardware board designed around an 8-bit Atmel AVR microcontroller, or a 32-bit Atmel ARM. Pre-programmed into the on-board microcontroller chip is a boot loader that allows uploading programs into the microcontroller memory without needing a chip (device) programmer. As written earlier, our main objective was to design a prototype and not a final product. Arduinois the world's leading open source electronic prototyping platform. Arduino has a number of support libraries and modules. One such module is the support for Bluetooth[8]. Hence our obvious choice.

C. Why Bluetooth ?

Bluetooth is a wireless technology standard for exchanging data over short distances from fixed and mobile devices, and building personal area networks (PANs). Bluetooth operates in the range of 2400–2483.5 MHz (including guard bands). This is in the globally unlicensed (but not unregulated) Industrial, Scientific and Medical (ISM) 2.4 GHz short-range radio frequency band. Bluetooth uses a radio technology called frequency-hopping spread spectrum. The transmitted data are divided into packets and each packet is transmitted on one of the 79 designated Bluetooth channels. Each channel has a bandwidth of 1 MHz.Bluetooth 4.0 uses 2 MHz spacing which allows for 40 channels. The first channel starts at 2402 MHz and continues up to 2480 MHz in 1 MHz steps. It usually performs 1600 hops per second, with Adaptive Frequency-Hopping (AFH) enabled.

There are several communication protocols in use, of which the most popular are Bluetooth, Infrared and Wi-Fi. Infrared is not as reliable as Bluetooth[9]. It can be easily shielded, and transmission can be interrupted easily. It is well on its way to becoming an obsolete technology. Wi-Fi uses radio transmission technology just like Bluetooth. Although Wi-Fi allows higher transmission rates, it consumes a lot of power, which is unsuitable for a wireless device. We need to send only small amounts of data. Since we have no need for higher transmission rates, we have chosen Bluetooth as our transmission protocol, thus eliminating the extensive power consumption factor. Nowadays, almost all devices support Bluetooth, and in the future if we plan to include other devices, then connecting them using Bluetooth is the only viable option. Bluetooth protocols simplify the discovery and setup of services between devices. Bluetooth devices can advertise all of the services they provide. This makes using services easier, because more of the security, network address and permission configuration can be automated than with many other network types.

D. Why HC-05 Bluetooth Module?

The HC-05 Bluetooth module[10] is readily available, can be easily paired with devices and is easily configurable. It is also cheap, compared to other available Bluetooth modules. It can easily be configured with the Arduino boards as well.

E. Why Amarino?

Amarino[11][12] is a toolkit to connect Android -driven mobile devices with Arduino microcontrollers via Bluetooth. The toolkit provides easy access to internal phone eventswhich can be further processed on the Arduino open source prototyping platform. Started as aproject at MITMedia Lab at the High-Low Tech group, this toolkit seeks to empower people to externalize their phone events to creatively demonstrate them on wearables, living spaces, or other tangibles.

We had a number of choices like ArduDroid and Sensoduino, which have a number of options that we didn't need and they weren't customizable. But mostly they did not have plug-in support. This is where Amarino comes in.Amarino has plug-in support which can be easily developed and customized by just knowing Android programming.

F. Why Nokia 5110 LCD Screen?

The ideal display device for this project would have been the Arduino TFT screen. This screen has SD card support, high resolution and full Arduino library support. But its limited availability forced us to look for an alternative.

The Nokia 5110 LCD[13] is readily available, data sheets can be easily found on the internet, has been tested with Arduino and also has relevant libraries present. The Nokia 5110 is a basic graphic LCD screen for lots of applications. It was originally intended for as a cell phone screen. This one is mounted on an easy to solder PCB.

It uses the PCD8544 controller, which is the same used in the Nokia 3310 LCD. The PCD8544 is a low power CMOS LCD controller/driver, designed to drive a graphic display of 48 rows and 84 columns. All necessary functions for the display are provided in a single chip, including on-chip generation of LCD supply and bias voltages, resulting in a minimum of external components and low power consumption. The PCD8544 interfaces to microcontrollers through a serial bus interface.

III. DESIGN

A. Connecting the Arduino Uno with PC:

The first step in the design phase of this project is to connect the Arduino microcontroller kit with the PC/Laptop using the USB (Universal Serial Bus) cable, as shown in Fig. 1. We use the Arduino IDE, installed on the PC/Laptop, to write code and communicate with the Arduino kit. The ArduinoIDE compiles and uploads the code written to the Arduinokit. When you upload a sketch, you're using the Arduinobootloader, a small program that has been loaded on to the microcontroller on your board. It allows you to upload code without using any additional hardware. The bootloader is active for a few seconds when the board resets; then it starts whichever sketch was most recently uploaded to the microcontroller. The bootloader will blink the on-board (pin 13) LED when it starts (i.e. when the board resets).

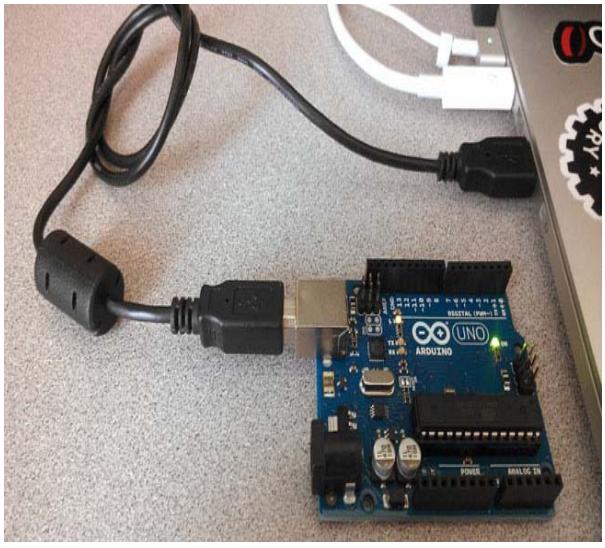


Figure 1. Connecting the Arduino to PC

B. Connecting Nokia 5110 LCD Screen With Arduino:

Now we connect the display unit shown in Fig. 2 with the Arduino kit in the following manner

Table 1.Connections between LCD and Arduino

LCD Pins	Arduino Digital Pins
Clk	3
Din	4
DC	5
RST	6
CE	7
LED	13

C. Connecting HC-05 With Arduino:

We use the HC-05 Bluetooth module (Fig. 3) with the Arduino kit to establish a Bluetooth connection which will enable communication between the microcontroller kit and phone.

The HC-05 Bluetooth module is the most economical and easiest way to go wireless (via Bluetooth). This module makes it easy for you to wirelessly extend your serial interface, so you can control any program running on your Laptop with serial port interface.

The 4 pins are:

VCC (Power 3.3 – 6V);

GND;

TXD;

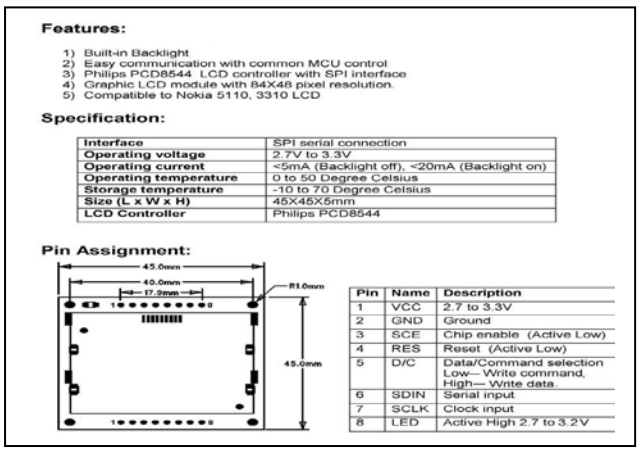
RXD;

Default pairing code: 1234

Default baudrate: 9600

Figure 4 shows how it's wired with an Arduino:

The RX and TX pins are used for receiving and transmitting respectively. Now we connect the RX/TX of the module to the TX/RX of the Arduino kit and V_{CC} and GND connections are made too.



Features:

1) Built-in Backlight
2) Easy communication with common MCU control
3) Philips PCD8544 LCD controller with SPI interface
4) Graphic LCD module with 84X48 pixel resolution.
5) Compatible to Nokia 5110, 3310 LCD

Specification:

Interface	SPI serial connection
Operating voltage	2.7V to 3.3V
Operating current	<5mA (Backlight off), <20mA (Backlight on)
Operating temperature	0 to 50 Degree Celsius
Storage temperature	-10 to 70 Degree Celsius
Size (L x W x H)	45X45X5mm
LCD Controller	Philips PCD8544

Pin Assignment:

Pin	Name	Description
1	VCC	2.7 to 3.3V
2	GND	Ground
3	SCE	Chip enable (Active Low)
4	RES	Reset (Active Low)
5	D/C	Data/Command selection Low— Write command, High— Write data.
6	SDIN	Serial input
7	SCLK	Clock input
8	LED	Active High 2.7 to 3.2V

Figure. 2. The Nokia 5110 Specifications

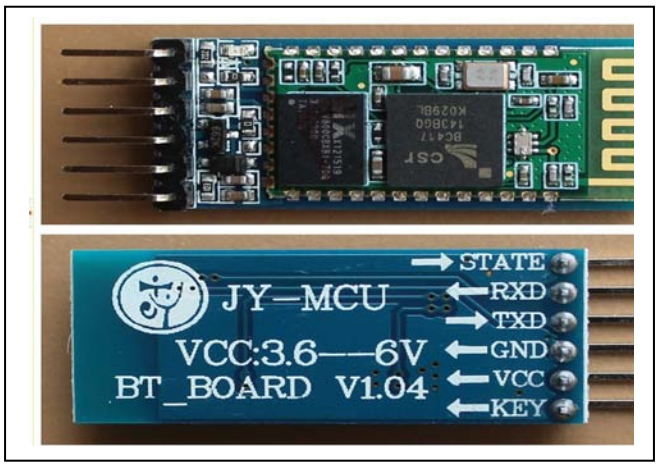


Figure 3. The HC-05 and its pins

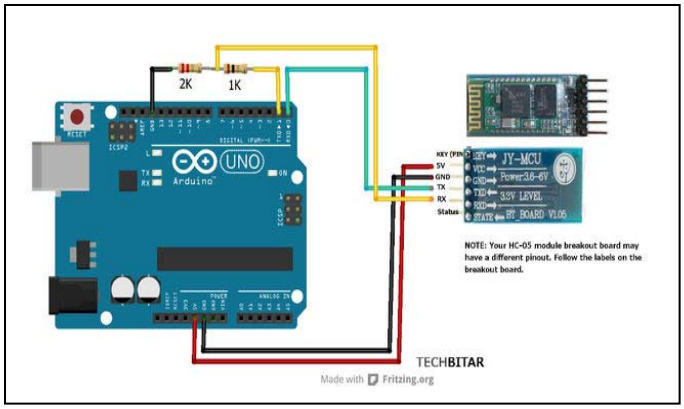


Figure 4. Connecting HC-05 with Arduino

D. Pairing Phone with HC-05:

Fig. 5 is the first screen we see when we launch Amarino on our phone. Since Amarino is all about connecting our phone to an Arduino, the very first step we have to do is to search for our Arduino Bluetooth device we want to talk to. To do that hit the "Add BT Device" button and wait until our Arduino Bluetooth module pops up. If it will not show up even if the discover process has already finished, we should check if your Arduino Bluetooth module is powered and discoverable.

We may find more than one device, as in Fig. 6, because of other Bluetooth devices around us. We need to find out which one is our Arduino Bluetooth module. After finding it, we select it to add to Amarino. (Each module has a unique address and normally also a human friendly name assigned).

If we have managed to add our BT device to Amarino, we see one new device, namely the device we have added, with a connect button next to it (Fig. 7) at the main screen of our Amarino application. We now have to pair (Fig. 8) the device with our phone. The default pairing number is '1234'.

After pairing, we press 'Connect' to connect the device with our phone. If connection is successful, then a green light lights up on the module. After connection is active, we proceed to add the respective events as in Fig. 9. We add events by clicking on the add event button.

Figure 5.Amarino opening screen

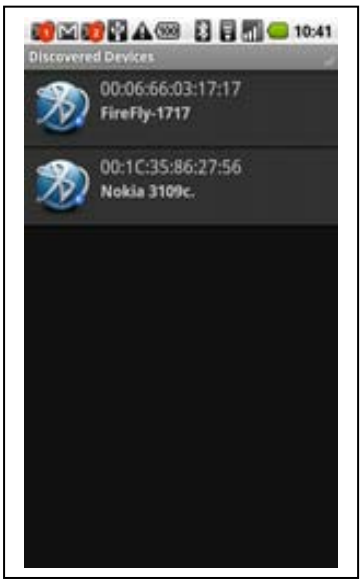


Figure. 6. Discovering devices

Figure. 7. Device added to Amarino

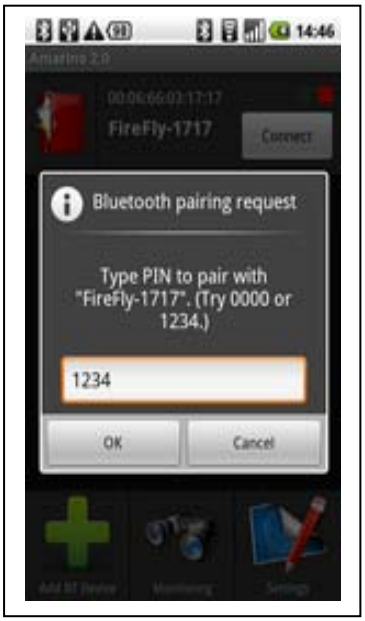


Figure 8. Pairing the device

Figure 9. Connect to device and add events

Finally we can monitor and send data to Arduinovia Amarino using the monitor option (Fig. 10). Finally, the event management module also gives us some feedback providing real-time data to us. Instead of monitoring, we go back to the event module (red cabinet icon) and see which random values are sent from the Test event. Real-time data are only visible if a connection is up and running. However we can force enable events to show their data without being connected (long press on an added event to get a context menu with options to force enable/disable).

Figure 10. Monitoring Amarino activity

IV. ACKNOWLEDGMENTS

We would like to extend our gratitude to our project guide Professor Romit Beed, whose constant guidance and recommendations made this project possible. We would like to thank Rana Biswas Sir whose enthusiasm, ideas and encouragement made us believe in our abilities and inspired us to pursue the idea of making this project a success.

All the teachers of the Department of Computer Science have helped us whenever we needed any. The lab staff has allowed us to use the facilities whenever needed.

All our classmates have been wonderful support during all the stages of this project. We would like to mention our friend ArjunilPathak for letting us use his Arduino throughout the development of the project. Last but not the

least, we are ever grateful to our families who are a constant support and inspiration in whatever we aspire to achieve.

V. REFERENCES

[1] http://www.android.com/

[2] http://www.google.com/about/

[3] http://opensource.org/

[4] http://www.forbes.com/sites/dougolenick/2015/05/27/apple-ios-and-google-android-smartphone-market-share-flattening-idc/2/

[5] http://www.apache.org/licenses/

[6] http://www.visionmobile.com/product/developer-economics-q1-2014-state-developer-nation/

[7] http://www.arduino.cc/

[8] http://www.bluetooth.com/Pages/Basics.aspx

[9] http://support.en.kodak.com/app/answers/detail/a_id/1685/~/bluetooth-versus-other-wireless-technologies/selected/true

[10] http://www.tec.reutlingen-university.de/uploads/media/DatenblattHC-05_BT-Modul.pdf

[11] Bonifaz Kaufmann and Leah Buechley. (2010). Amarino: a toolkit for the rapid prototyping of mobile ubiquitous computing. In Proceedings of the 12th international conference on Human computer interaction with mobile devices and services (MobileHCI '10). ACM, New York, NY, USA, 291-298.

[12] Bonifaz Kaufmann. (2010). Design and implementation of a toolkit for the rapid prototyping of mobile ubiquitous computing. Master's thesis. Alpen-Adria-Universität Klagenfurt, Klagenfurt, Austria.

[13] https://www.sparkfun.com/datasheets/LCD/Monochrome/Nokia5110.pdf