Självständigt arbete på grundnivå

Independent degree project - first cycle

Dateknik

Computer Engineering

Using Blockly to Create Simple Sensor & Actuator Based Applications on the SensibleThings Platform

Yuanhong Xia
MID SWEDEN UNIVERSITY
Department of Information Technology and Media

Examiner: Ulf Jennehag, Ulf.Jennehag@miun.se
Supervisor: Victor Kardeby, Victor.Kardeby@miun.se, & Stefan Forsström, stefan.forsstrom@miun.se
Author: Yuanhong Xia, yuxi1001@student.miun.se
Degree programme: Internationl Bachelor’s Programme in Computer Engineering, 180 credits
Main field of study: Computer Engineering
Semester, year: VT, 2014
Using Blockly to Create Simple Sensor & Actuator Based Applications on the SensibleThings Platform
Yuanhong Xia
2014-08-02

Abstract
Research at Miun investigates the so called “Internet of Things”. The research has so far produced components for sharing sensor and actuator information on the Internet, most notably the Sensible Things platform. The Sensible Things platform can run on many different devices such as smart phones, raspberry Pi devices, and desktop computers. However, programming applications based on the Sensible Things platform and deploying it on the hardware require knowledge of the Java programming language and the API of the platform. Today, many novice and first time programmers learn to program using the Blockly programming method, which include connecting blocks together to form simple procedures. Therefore, we have applied the Blocky method to ease the development of simple applications on the Sensible Things platform. After the work was done, the general users are now able to create entry level Sensible Things Applications.

Keywords: Google Blockly, Java, JavaScript, Internet of Things, Sensible Things platform
# Table of Contents

Abstract.......................................................................................................................... iii

Terminology / Notation.................................................................................................... vi

1 Introduction.................................................................................................................. 1
1.1 Background and problem motivation................................................................. 1
1.2 Overall aim............................................................................................................. 1
1.3 Concrete and verifiable goals............................................................................... 2
1.4 Scope...................................................................................................................... 2
1.5 Outline.................................................................................................................... 3
1.6 Contributions......................................................................................................... 3

2 Theory.......................................................................................................................... 4
2.1 Internet of Things................................................................................................. 4
2.2 Sensible Things Platform.................................................................................... 4
2.3 Sensible Things Platform Application............................................................... 5
2.4 Visual programming language............................................................................. 5
2.5 Google Blockly platform.................................................................................... 6
2.6 Google Blockly Application............................................................................... 6
2.7 Closure Library..................................................................................................... 7
2.8 JsZip and fileSaver Library................................................................................ 7
2.9 Raspberry Pi......................................................................................................... 7

3 Methodology................................................................................................................. 8

4 Design / Implementation............................................................................................ 10
4.1 Google Blockly Application Design..................................................................... 11
4.1.1 Blocks Design.................................................................................................. 11
4.1.2 Code Generator............................................................................................... 12
4.2 Sensible Things Application Design.................................................................... 14
4.2.1 Sensor and Actuator Classes.......................................................................... 15
4.2.2 sensible things platform handler.................................................................... 15
4.3 File Generation..................................................................................................... 18

5 Results.......................................................................................................................... 19
5.1 Blockly Application............................................................................................. 19
5.2 Blockly Application Code Generation............................................................... 20
5.3 Download Files..................................................................................................... 21
5.4 Running the Sensible Things Applications......................................................... 21

6 Conclusions.................................................................................................................. 23
6.1 Understanding blockly......................................................................................... 23
6.2 Sensible Things Platform Application............................................................... 23
6.3 Design blocks......................................................................................................... 23
6.4 Export ready-to-use Files.................................................................................... 23
6.5 Create a proof-of-concept application with Blockly App................................. 24
6.6 Evaluate the end results and propose future work............................................. 24
Terminology / Notation

Acronyms/Abbreviations

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>IoT</td>
<td>Internet of Things</td>
</tr>
<tr>
<td>Miun</td>
<td>Mid Sweden University</td>
</tr>
<tr>
<td>VPL</td>
<td>Visual programming language</td>
</tr>
<tr>
<td>UCI</td>
<td>Universal Communications Identifier</td>
</tr>
<tr>
<td>DCXP</td>
<td>Distributed Context eXchange Protocol</td>
</tr>
</tbody>
</table>
1 Introduction

Broadband Internet has been becoming widely spread. The cost of it is decreasing rapidly, the number of devices that support it is increasing dramatically, the investigation for the development of it is boosting. All these facts are contributing to the scenario in which everything is connected. The idea of this scenario came out long time ago and the universal recognized term, 'Internet of Things', was proposed in 1999[1]. The simplest example of IoT is a switch connected to the Internet and can be switched by another device remotely.

Miun has been investigating an open source platform called “Sensible Things”. It is a platform which enables fast and efficient development of IoT applications[2]. IoT is the future, There is a huge amount of data transferred in networks, the most common way to store and handle them is by using database. It is a sophisticated technology but it does not scalable with the growth to the worldwide IoT and it is too centralized which is not as reliable as distributed system. Sensible Things applies peer-to-peer technology which eliminates all these drawbacks. Besides, this platform is supported by many devices including PC, android smart phones and raspberry Pi devices. The platform can also be applied in many fields, e.g., smart home, health care monitor and so on. Nevertheless it relies on its own API and requires the knowledge of Java programming, The requirements might be relatively high to general users and programming beginners.

1.1 Background and problem motivation

Sensible Things Platform is really an easy-to-use platform, but it could be easier. Google Blockly is a web-based, visual graphical programming editor. Users can build applications by simply joining blocks without the knowledge of coding, which means Google Blockly can be regarded as a high level graphical programming language. In order to lower the entrance to general users, the developers want to apply the google blockly environment to the sensible things platform. Hardware on the other hand should also be considered and Raspberry Pi is an ideal one. It is essentially a bare-bones personal computer with a Linux operating system installed[3]. 30 years ago, many people could not imagine that PC would be so popularized. And now users require more, it should be powerful as well as cheap. Raspberry Pi does more than that, it is also tiny with low power consumption. As the Linux is installed, Java is well supported.

1.2 Overall aim

The problem of this project is to implement a method for building simple sensor and actuator applications by using the Blockly programming blocks. In detail,
Using Blockly to Create Simple Sensor& Actuator Based Applications on the SensibleThings Platform  
Yuanhong Xia  
2014-08-02

one should be able to create sensor logic in Blockly and then export a ready-to-use package to be applied directly on the hardware. The difficulty of making such a project is indefinite since Google Blockly is a quiet new technology and it lacks of instruction manual, tutorial assistance and forum discussion. The main purpose of this project is to make a tool for the development of Sensible Things platform rather than enhancing the platform itself, so studying blockly is of the highest priority. Nevertheless, to make this tool without the knowledge of the platform is impossible which means the author should get the understanding of the platform at the same time. After these works done, the main work will be designing the specific blocks and java applications so that the Java code can be divided into small pieces and match to each block.

1.3 **Concrete and verifiable goals**

To solve all those problems stated in the previous section, this project is solved by following the manner so called “divide and conquer”. It is separated into 6 specific goals listed below:

1. Make and test the corresponding application in tradition Java coding way.

2. Design blocks for sensible things platform to ensure that the final code it generates is valid.

3. Make it possible to export the code into a ready made packages for both desktop computers and Raspberry Pi devices

4. Create a proof-of-concept application using a sensor, an actuator, and a console application by using Google blockly

1.4 **Scope**

This project is limited to building the Google Blockly application and the application made from blockly. It does not include any security problems, add-in building or the development of the Sensible Things platform itself. The study of blockly and the blocks design are the main focus as the only way to learn it is by reading the source code. Because the author has almost no knowledge to the field of security, so the security problem is out of the scope. And since the Sensible Things platform is developed into a mature stage, the author will not try to improve it, but an application made with the platform is essential. The blockly application is an off-line applications, so the sever side programming is out of the scope.
Using Blockly to Create Simple Sensor & Actuator Based Applications on the SensibleThings Platform
Yuanhong Xia

1.5 Outline
In Chapter 1, the overall introduction of project is described briefly to provide the reader a brief idea of the whole report. The theory work will then be stated to in Chapter 2. Then Chapter 3 presents the main methodology applied in this project, demonstrating the detailed composition. In Chapter 4, the concrete design and implementation are presented and the results will be shown in Chapter 5 and all the implementations and results will be discussed and analyzed respectively. Finally, some evaluations about the project is made in the laste chapter to provide the acknowledgment of the author to this project.

1.6 Contributions
The whole work is conducted solely by the author. The Google Blockly platform is developed by Google Blockly team and the Sensible Things Platform is created by Miun. The coding/implementation part for the features of this project is performed by the author.


2 Theory

In this chapter the terms and theoretical background to help understand the rest of the report are presented. The knowledge provided here is very project based and it is highly suggested that the reader should have some basic knowledge of network, JavaScript, closure library and Java.

2.1 Internet of Things

The Internet of Things is a network of physical computing-like devices. All these devices must be assigned unique identifiers and they should be embedded with the technology to transfer data within the network without participation of human. In other words, devices can feel and communicate.[4] The figure 2.1 below demonstrates a simple IoT application. The plant can send messages to twitter according to the humidity of the soil.

![Figure 2.1 A plant uses Twitter](image)

2.2 Sensible Things Platform

The Sensible Things platform is a distributed system based platform which has been divided into 5 different layers as shown in figure 2.2. The interface is a
Using Blockly to Create Simple Sensor & Actuator Based Applications on the SensibleThings Platform
Yuanhong Xia

2014-08-02

A generic interface which enables developers to create Sensible Things application based on this platform. The add-in layer allows developers to add additional functionality to the platform. The dissemination layer is the layer which handles all the communication mechanisms and protocols such as DCXP. The network layer addresses all the underlying network architecture. And the sensor and actuator layer abstracts all the sensors and actuators connection to the platform.

2.3 Sensible Things Platform Application

The image 2.3 show how applications in this platform work. After registering all nodes in the platform the data-seeking node should resolve the data-source node first and the source will return the node information. Then the data-seeking node can use “get” along with the uci and node information to trigger the “getEvent” method of the source end. In the function body of “GetEvent”, there is the method called “notify” which will pass the value back and can be retrieved in the function “GetResponse”. This is the way to get sensor value. And if one node needs to set the value of another node it should use “Set” to trigger “SetEvent” of the other node which can pass the value. It is obvious that calling “Get” and “Set” functions always requires UCI And NODE information.
Using Blockly to Create Simple Sensor & Actuator Based Applications on the SensibleThings Platform
Yuanhong Xia

2.4 Visual programming language
Visual programming language is a language which allows the users to program graphically rather than specifically textually. Since VPL is an abstract concept there is no restrict rules to define whether a language is a VPL. There exist multiple VPLs, among which Scratch, Alice, Koudu and Google Blockly are the most famous ones. But each of them are build for different purposes and based on different mechanisms.[7]

2.5 Google Blockly platform
Google Blockly is a graphical programming language made up of blocks which isolates users from traditional syntax errors.[8] It is a relatively new VPL but it is quite easy-to-use. The figure 2.5 briefly demonstrates how this platform works. Users can drag and joint blocks in the workspace, each blocks generates a piece of code and depending on the type of the block, their code can be retrieved by other blocks by using different function calls. Then the workspace can use a function call named “workSpaceToCode” to ingredient all those codes together. The platform is programmed by using JavaScript and Closure Library.
Using Blockly to Create Simple Sensor & Actuator Based Applications on the SensibleThings Platform
Yuanhong Xia

2.6 **Google Blockly Application**

The applications are web-based. Each application should have an individual folder placed under the apps folder. There should be an html file which includes all the general JavaScript files and a template file. The template file includes all the application specified JavaScript Files and libraries, such as self defined blocks file third party libraries. In the template all the html tags that constructs the page are located and it should also arrange the blockly workspace and blocks toolbox. [9]

2.7 **Closure Library**

The closure library is a JavaScript library which is broad, stable and cross-browser. Comparing to jQuery, the library is rich and the styles of the interfaces are united so it is easier to maintain and update. Besides the closure library can greatly compress the JavaScript Code which dramatically increases the efficiency. Many Google products such as Gmail, Google Maps and Google
2.8 **JsZip and fileSaver Library**

These two libraries are third party libraries built for specific usages. FileSaver.js is a perfect solution for client-side file generation and downloading. JsZip.js on the other hand helps to organize all files required and compress them into a zip file since no browser supports download multiple files with a single action.[11]

2.9 **Raspberry Pi**

A Raspberry Pi is a tiny computer which can be connected to monitor, keyboard and many other peripherals. The size of it is similar to a regular student card. But it is powerful as it can do many things a regular PC does. It ingredients 256 MB or 512MB memory and uses Memory Card as secondary storage. It installs Linux as operating system which also means it supports Java. And it supports network connection so it is fully qualified for this project. Besides, it is inexpensive.[12]
3 Methodology

Generally speaking, the strategy for fulfilling this project is to divide and conquer. The whole project is aimed at applying Blockly environment onto the development of Sensible Things application, which means research on Blockly and Sensible Things are important. After getting enough knowledge of these platforms, the author will make some effort on trying to create some blocks and sensible things applications. After that, the author will design and arrange blocks which are reasonable to generate code for applications. The last job is to debug and polish.

The first goal is to study google Blockly, usually the study of a language or a platform is not a concern in a report, but Google Blockly is quite a new environment which lacks of tutorial and instruction manual. So the study of it is vital to this project, and which takes up a long time to fulfil. The utilization of blockly will be described in the design chapters.

The second goal is to make an Sensible Things Application, the author needs to read the related tutorial and test the example application. The code for this platform is written in Java which is organized in 5 layers, within which add-in layer is out of the scope. The goal of this project is to enable users making sensible things application without the knowledge of itself, therefore, the author made an application first by following the instruction from the official website. The final application build for this project is a basic one. Since Java application is famous for its platform in dependency, the programming and debugging were tested on PC. Then it was tested on Raspberry Pis after it became reliable enough. As the sensor and actuator are jointed on Raspberry Pi, The applications running on pc used simulated values and action.

The second goal is specific blocks and blockly application design and implementation. In order to make it the codes matched to blocks, the code should be organized and divided into small pieces. And for the purpose to lower the entrance to the general users, the mechanism of Sensible Things platform and the specific logic to handle sensors and actuators are hidden in fixed classed, users can directly call and use. The application code is organized based on the blocks the author designed, and the blocks should be modified with the requirements of the code, these operations should be taken concurrently. Sensible Things platform does offer the function to subscribe sensor but it is too complicated. So this application applies the most inefficient method: polling. To make it easier for users to interact with blockly, some blocks must be dynamic, i.e., the uci of node that users declare with the set-node blocks should be in the drop-down list of the get-node blocks.

The forth goal is making it possible to export the code generated, to achieve which the author should create a button, by clicking which, the generated codes
should be separated into multiple files, each of which is for one device. And these codes should be stored in java files which are going to be compressed into one zip file and the zip file should be able to download. To make the best of browser compatibility, the application ingredients jszip library, blob library and filesaver library.

The fifth goal is to use the blockly application create a sensible things application. The author will make it purely with Blockly application without writing a single java statement. Then the procedure of debugging and polishing comes last. To be specific, the blockly application should be bug-free and the java application made from it should also run without bugs. The application will be tested both on PC and raspberry devices. Besides, the blockly application should also be user-friendly, the author will try to add tool-tips to blocks, offer instructions and so on.

The last goal is to evaluate the end results and propose future work. There are many visual graphical development environment, such as Scratch, Alice, Koudu, etc. The author will attempt to study each of them and make the comparison. Different web browsers have different compatibility for JavaScript libraries, in order to make the blockly application well supported, the author will test it on several browsers including IE, Firefox and Chrome.
4 Design / Implementation

This chapter demonstrates the structural design of the solution. Figure 4.0 below provides the graphical design solution of this project.

When the users have some ideas like: when the temperature of living room is higher than 30 degree, open the window. They can use this blockly application to drag and joint blocks to achieve that without the knowledge of programming. And the blockly application will generate application code file for each devices. The devices must support Java though. Then users can simply follow the instruction and run those applications.
4.1 Google Blockly Application Design

The application is written in JavaScript and is divided into three layers: The blocks layer, the code generator layer and the core layer. The core layer manages all the underlying methods such as “valueToCode”, “statementToCode”, etc, and this layer is the most dedicate layer. This project does not modify any of this layer as it plays the role of the basic compressed library. The blocks layer manages the shape and other attributes of blocks and the code generator layer handles the code each blocks generates.

4.1.1 Blocks Design

In order to make the blockly application suitable for Sensible Things applications, the blockly application utilities 10 blocks as shown in figure 4.1.1: Application frame block, Sensor block, Actuator block, monitor block, resolve block, node block, get block, set block, print block and blank block. Since the way to design a block are fixed, this chapter just chooses some typical blocks to present.

![Figure 4.1.1 self-defined blocks](image-url)
Using Blockly to Create Simple Sensor & Actuator Based Applications on the SensibleThings Platform
Yuanhong Xia 2014-08-02

The figure 4.1.2 below shows the sensor block, each text input area has a default value in it and bounded with a unique ID. The value input can be retrieved by the code generator. And some fields are fixed for the developer may offer limited choice to the general user, take the model field as an example, this filed decides which sensor actuator class should be declared.

The figure 4.1.3 demonstrates the monitor block. As mentioned before the application applies polling to monitor the values of sensors and this can create a control statement which loops for every certain time. The jigsaw-shape area can join a value block such as numbers or variables and the other area takes statement block only.

4.1.2 Code Generator
First of all, the built in Google Blockly generator does not support Java code generating, the author must write some script for Java supporting. In the folder “generators”, a Javascript file named “java.js” should be created first. Inside the file, all the reserved key words should be registered and it handles the blocks, workspace and collection management. Then a folder named “java” should be created, inside which specific Java code generating script files for reserved blocks should be created. All these files should be programmed by JavaScript and Closure library. In addition, the most important blocks are the 10 self-defined blocks:
Using Blockly to Create Simple Sensor & Actuator Based Applications on the SensibleThings Platform

Yuanhong Xia

2014-08-02

The image 4.1.2 shows how a block generates code, this is the application_frame block. It has one text field and three statements field, within which multiple corresponding blocks. Each field and has a unique ID and the code the sub blocks generates can be retrieved by using method "statementToCode" with the field ID. At the end, the block can return the code it is supposed to generate. Since it has no connections, it can not pass code to another block which means it can only be used as the outermost block and the code it returns should be retrieved by the workspace with the method "workspaceToCode". Each application block and the blocks within is for one device. All the code generated by other blocks will be placed in the main method.

The image 4.1.3 shows the dynamic blocks user interface.
And the figure 4.1.3 demonstrates the sensor block and the actuator block. The way to generate code is the same to application frame block. But there are three different features. One is that they have connections so they can pass code to another block. And the other one is that it has a drop-down list ("Model") since that sensor and actuator the devices support are fixed and depending on the choice of the user, the class declaration and implementation will be set. The last one is that they implement "getVars" function which will update the uci information to a global array which the application creates when it is initialized. And it can be retrieved by any other block without connection. Take the Node block as an example. It uses "FieldVariable" function call to retrieve the data that the sensor and actuator blocks updated to the global array. So the drop down list of this block is dynamic which will change with the modification of the sensor and actuator blocks.

4.2 Sensible Things Application Design

The application is divided into three layers: The operation layer, the sensible things platform handler layer and the sensor-actuator layer. The sensible things platform handler layer and sensor-actuator layer are hidden from the general users as these layers requires understanding and API for the Sensible Things platform, and as mentioned in the introduction chapter, the purpose is to lighten the burden for the general users. Users just need to call the sensors and actuators they need to install and make some logic for them.

Figure 4.2 Sensible Things Application work flow chart
It is the most complicated scenario as shown in figure 4.2 that the sensor, actuator and logic are all distributed in different devices. The simplest one is they are centralized in one node which does not require sensible things platform at all. The application applies polling to monitor the value of the sensors since it is the most implementable way. The Sensible Things platform does offer the methods to subscribe sensors but it requires knowledge out of the scope of the tutorial. There could be multiple sensors and actuators, so each sensor is monitored by one thread to use infinite loop to monitor. The polling thread is located at the logic node. So if users have just two devices, they can configure one works both as sensor node and logic node any other combination.

4.2.1 Sensor and Actuator Classes

There could be many models of sensors and actuators and each one requires to implement a class. Since the author currently only has one temperature sensor and one light switch actuator so only two sensor and actuator classes, one for each respectively. All classes should extends the class named “SensorActuator” from the Sensible Things platform library.

The class “Temperature” is a sensor class which implements the “getValue” method by reading the local file “temp.txt” and retrieve the data. Since the text written in the file consists of three parts separated by white-space characters and only the last part is required, the method splits the string into an array and retrieves the last entry.

The class “LightSwitch” is an actuator class which implements the “setValue” method by sending a terminal command line. Sending “gpio write 1 0” to turn on the light and “gpio write 1 1” to turn off it. When test this on windows platform it will catch errors since these are not valid commands for windows command line, but the statement to print messages can be used to test the result.

4.2.2 sensible things platform handler

This layer contains only one class, “PlatformHandler”. In order to make sure that the platform works proper and efficient, the author implements four maps as shown in figure 4.2.2. Each of the maps uses the uci as key since an uci is a unique identifier. The first map matches ucis with sensors and the second one matched ucis with actuators. When users implement a sensor or actuator, it will be added into a corresponding map, and after the implementations of sensors and actuators, a declaration of the platform handler is required whose constructor contains two arguments which are two maps so that the two maps can be passed into the handler. After the construction the platform will register each of the ucis. Since there might be multiple sensors and actuators and more than one thread to monitor the values, the handler should match the value with the uci in another map to avoid confounding of data. And the node information is also needed when use “SET” method, so the handler requires one more map to match ucis and nodes.
As mentioned in chapter 2.1, if one end calls the resolve function, the “resolveResponse” function will be called. In the body of this function, the function “Get” is called which will trigger the “GetEvent” function in the other end. The figure below demonstrates the specific implementation of these methods. It is shown that the function analyzes if the uci exists first. There might be three situations:

1. the uci belongs to an existing sensor, then it will return the value of the sensor;
2. the uci belongs to an existing actuator, then it will return nothing;

3. the uci does not exist at all, than it will tell the other end that it does not exist.

And one end can also use “Set” method to trigger the “SetEvent” method of the other end. And simillar to “GetEvent”, it will analyses the uci first. If the uci belongs to an existing actuator, it will call the “setValue” method of the actuator. The figure 4.2.3 demonstrates how this works.

Figure 4.2.3 How the polling thread sets the actuator value
4.3 File Generation

The figure 4.3 represents how the files are organized. The code generated from the workspace is a single String but the codes are designed for multiple devices. In order to divide them into separated files each application block has a comment in the end and the whole String can be spilted by that comment. And the code will be stored in individual java files and each file is named as the class name. Then all the files are going to be compressed in a zip file along with the essential jar file. The project is a offline blockly application so all these procedures are handled by client side alone which means it is hard to load the local file. So the author must convert the jar file into base 64 binary code and store it in the source code then turn it into jar file when users download it. Then the users can use the command line or terminal line to compile and run the applications. (javac -cp ./pi.jar;SensibleThingsBeta4.jar *.java and java -cp ./pi.jar;SensibleThingsBeta4.jar FILENAME)

Figure 4.3 Files organization
5 Results

This chapter demonstrates the results of this project which includes the blockly application, the code generation, the file generation and the sensible things applications.

5.1 Blockly Application

The picture 5.1 describes how to solve the scenario mentioned in chapter 4.2. Since there are three nodes, the system requires three devices and one application for each one. So the author dragged three application blocks, one for each device. The two in the bottom are for sensor node and actuator node respectively. And the one above is used for logic handling. It will fetch the value of the sensor every 6 seconds and if the value is greater than 0 it will turn on the light. Here is the advantage of using Google Blockly application, the users do not need to know the syntax of a language, they just need to have the ideas and joint the blocks. The whole process just looks like a plain language description.
5.2 Blockly Application Code Generation

Figure 5.2 demonstrates the code generated by logic application block and all the blocks connected to it. All the code in gray cyan part is generated by the application block and the others are generated by blocks in the logic part. The codes in the gray cyan part include all the import statements, the main class the main method and platform handler declaration, The monitor block generates a new thread which uses polling to monitor the value. The resolve block generates code to resolve the node. And the other logic blocks generate the codes in the if control statement. All codes are well organized.

As mentioned in chapter 4.2.2, each application requires 4 maps, and 2 of them are used to record the ucis of sensors and actuators. These two maps are declared in the main class. And when a Sensor or an Actuator block is joint in the application block, it will generate the code for declaring a sensor or an actuator
and letting the map to add it. The image 5.2.2 shows the code generated by the application block connected with a sensor block.

![Figure 5.2.2 code generated for the sensor node](image)

### 5.3 Download Files

After clicking the download button the codes will be divided and stored in individual java files, and All files along with the pi.jar file will be compressed into one ready-to-download zip file. The name of the zip file is always “files.zip” but the java files are all named by their class names. These java files can be easily compiled. And then there will be no problem to run them in corresponding devices. The figure 5.3 shows the view of the downloading package.

![Figure 5.3 file package](image)
5.4 Running the Sensible Things Applications

The figure 5.4 presents how the result of the sensible things applications look like when they are running. The scenario consists of three devices but the sensor node does not print out any information. So the screen shots below just demonstrates the actuator node and the logic node. Since the author set the condition to greater than 0 and the temperature is always above that level, the logic node constantly sends the command to the actuator node to turn the light on. And the author has tested many scenarios and all of them work well.

Figure 5.4 screen shots of the running sensible things applications
6 Conclusions

The blockly application created within this project indicates that the possibility to simplify the development of sensible things application exists. The overall purpose of this project is reached and the reasons why it is considered successfully are provided below.

6.1 Understanding blockly

This goal was fulfilled first. The author studied and gained enough knowledge and understanding of the Google Blockly platform. This is conducted by reading the limited resources online, reading the source code of official applications, making modification and testing the source code, understanding some third-party projects code. And the knowledge gained here was successfully applied to some other parts of this project.

6.2 Sensible Things Platform Application

This goal was successfully achieved. The author had read through the document provided by the sensible things official website and tested the example code on the page. Further, the author finished the exercises on the website. After all these preparations were done, The author designed his own application code which suits the requirement of the project. In addition, the author tested the applications many times on different devices and made sure that it is flawless. And the codes were organized into parts and classes which is prepared for dividing into individual pieces so that they can be assigned to blocks.

6.3 Design blocks

This goal was successfully completed. All ten blocks are delicately designed as shown in chapter 4.1.1. All of them are designed to generates some codes and some of them have drop box with options. With different combination of choice, these blocks will generate different codes. And the options of the node block are even dynamic which will change with the modification of sensor and actuator blocks. Among all these blocks, only one block is logically useless which is the blank block. This one has only a static text on it which is “Do nothing” and this block does not generate any code. Its existence is just to make more sense to the beginners. With different choice of blocks, different combination and different option selection in each block the blockly application can generate java files satisfying different scenarios.

6.4 Export ready-to-use Files

This goal was reached successfully. Instead of using the build in JavaScript functions, the author includes three third party libraries: “blob.js”, “fileSaver.js” and “jszip.js”, but the blockly application can not generate a jar file directly. There are 2 reasons that the author chose this way: The first one is that this is a client-side only application, for security reason, the JavaScript command can
not call cmd or terminal lines directly and it can not compile java files directly. The second reason is that by using these existing libraries, the application can achieve best compatibilities for the browsers since each of these libraries are designed and tested by the original creators delicately.

### 6.5 Create a proof-of-concept application with Blockly App

This goal is also conducted successfully. The author had tested the sensible things applications generated by blockly application for different scenarios. And they were tested on different devices also which include windows 7 installed PC and raspberry Pi. In all these cases the applications work fine

### 6.6 Evaluate the end results and propose future work

There are many visual graphical development environment, such as Scratch, Alice, Koudu, etc. The reason why the author chose Google Blockly is because it is open source and enough for code generating. Although it only supported to generate JavaScript, Python and Dart code, the author managed to make it generate proper Java code for this project. Another good feature of Google Blockly is that it is JavaScript based, which enables easy insertion into web pages and web applications. Sensible Things is relatively a new technology but it is really easy-to-use since it has hidden all algorithm for network protocols, distribute system from the user, which dramatically lightens the burden for programmers who are not specialized in these fields. In the future the blockly application can be expanded to satisfy more requirements. For example, developers can add date block which will generates code to make function calls to get the current time and make logic comparisons. And the developers can build the server side applications which will eliminate all the drawbacks mentioned before. And not only the blockly applications can be applied to sensible things platform, it can also be applied to other technology which can lead beginners to get start easier and faster.

### 6.7 Ethics

As mentioned in the scope section, the security and authentication are not part of the concern of this project. In this case user can not keep their sensor data private, anyone knows specific ucis can access to the corresponding data. As users of this blockly application, they should keep the private information as classified as possible to avoid others’ intrusion. The purpose of this project is to help lower the entrance for general users and to make Sensible Things platform known by more developers. Anyone abused the code generated by this blockly application are considered malicious and such actions should be prohibited.
References


[2] SensibleThings
http://sensiblethings.se/about.html

http://recruiterbox.com/blog/whats-the-big-deal-with-raspberry-pi/


[5] Challenges when Realizing a Fully Distributed Internet-of-Things
– How we Created the SensibleThings Platform
Stefan Forsström, Victor Kardeby, Patrik Österberg, and Ulf Jennehag
Retrieved 2014-07-021

The SensibleThings Platform Stefan Forsström


https://code.google.com/p/blockly/wiki/FAQs

https://code.google.com/p/blockly/wiki/Tutorial_CreateYourFirstBlock

https://developers.google.com/closure/library/?hl=zh-CN
Using Blockly to Create Simple Sensor & Actuator Based Applications on the SensibleThings Platform
Yuanhong Xia


https://github.com/eligrey/FileSaver.js/
