12-2016

# The Eco-Smart Can

Darack B. Nanto
*East Tennessee State University*

2016

# The Eco-Smart Can

DARACK NANTO

NOVEMBER 2016

# The Eco-Smart Can

By
Darack B. Nanto
November 2016

An Undergraduate Thesis Submitted in Partial Fulfillment
of the Requirements for the
University Honors Scholars Program
Honors College
and thek
Honors-in Manufacturing Engineering Technology Program
Department of Engineering Technology, Surveying and Digital Media
East Tennessee State University

| | |
|---|---|
| Darack Nanto | Date |
| Dr. Paul Sims, Thesis Mentor | Date |
| Dr. Moin Uddin, Reader | Date |
| Dr. Daryl Carter, Reader | Date |

# ABSTRACT

The Eco-Smart Can

By
Darack Nanto

I noticed that maintenance workers had the same itinerary when emptying trashcans, meanwhile some trashcans needed to be emptied urgently. Traditionally, ETSU maintenance operate on daily routes to pick trash on designated time, regardless the level of the containers. The time, resources and labor invested in collecting the trash could be saved. Therefore, I decided to use the Internet of Things (IoT) to create a device that will optimize trash collection, to reduce costs and pollution.

# ACKNOWLEDGEMENTS

I would like to thank God, my family and my friends, especially Ms. Oceane Tanny, for supporting me during this process. They helped me in so many aspects. They suggested some interesting ideas that sometimes helped me overcome some difficulties.

I would like to say a special thank you to Dr. Paul Sims for mentoring me during this project. I would like to thank him for sharing some valuable knowledge that helped me solve critical problems when I was developing my prototype. In addition, I would like to thank Dr. Moin Uddin and Dr. Daryl Carter for being my thesis readers.

I would like to thank the ETSU Honors College and ETSU Sustainability Fee, for giving me funds to purchase all required equipment to conduct my research. A special thank you goes to Ms. Kathleen Moore Director of the Department of Sustainability for sharing some information concerning the mode of operation of the ETSU Maintenance Staff work and she put additional resources at my disposal that helped me complete this project.

I would also like to thank Dr. Keith Johnson our Department Chair, for advising me and making sure I had access to resources I needed in the Engineering Technology Department.

Finally, I would like thank everyone that I did not mention that helped me directly or indirectly for the accomplishment of this thesis.

# DEDICATION

I dedicate this thesis to my family and friends. In addition to the ETSU Maintenance Staff that works hard every day to keep our campus clean and beautiful.

# Table of Contents

# 1. Overview

With an increase in the population of East Tennessee State University's (ETSU) community and the creation of new football stadium, the cleanliness of the campus needs to be maintained and improved. Also, keeping the campus clean will help prevent some diseases, amongst which some caused by mold. Traditionally, ETSU maintenance operates on daily or biweekly routes to pick up trash and recycle bins on a designated time, regardless of whether the containers are full or not (Fig. 1-1 shows a schematic diagram of such un-optimized system).

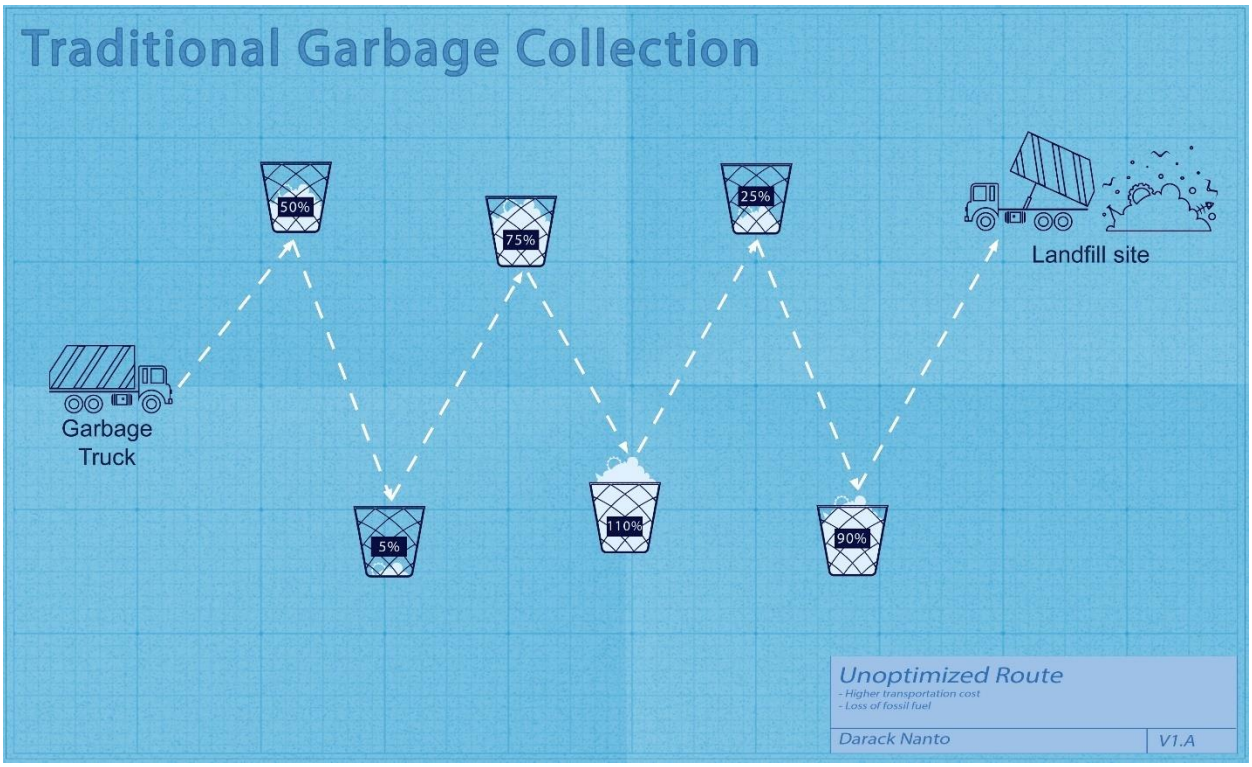



*Fig 1-1: Un-optimize Trash collection route.*

Time, resources and labor combined in collecting the trash could be saved if the ETSU maintenance knew which trash needed to be empty at the right time. Therefore, I decided to use the Internet of Things (IoT) to create a device that will optimize trash collection (Fig. 1-2), to reduce costs and pollution.

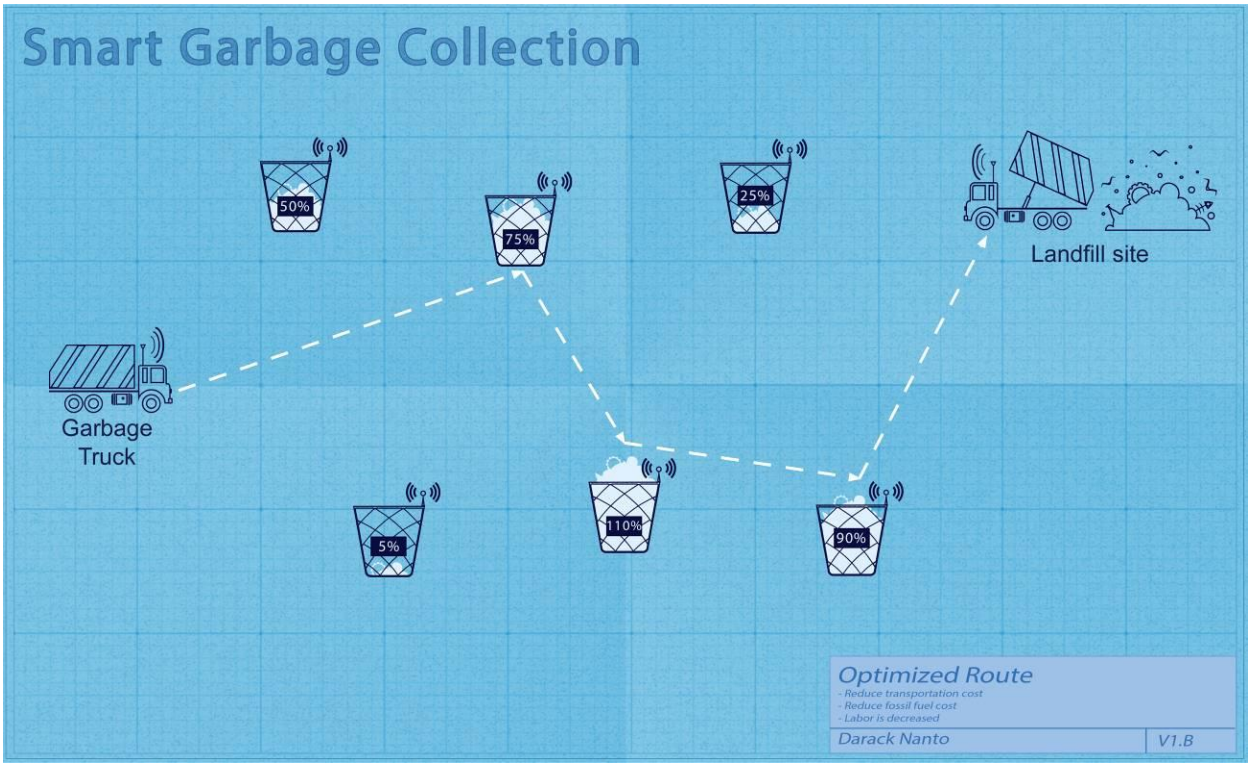



*Fig 1-2: Optimize Trash collection route.*

The IoT is the concept of connecting any device or man-made object to the internet. It provides the ability to transfer data over the internet. My project, “The Eco-Smart Can”, aims at using the same concept of the IoT and connecting a built device using an open-source computer and software to send data from a traditional trash container to the maintenance facility office. The objective of this project is to create a device that will shoot sonar waves to know the level of the trash in a container. It will also measure temperatures inside the container, because high temperatures can cause bacteria or germ to reproduce faster. Data collected from the sensors will be sent over a cellular network General Packet Radio Service (GPRS) or the Internet (through Wi-Fi) for analysis and displayed on Ubidots which is a cloud web platform to display collected data as shown in Fig. 1-3. I will set-up the platform in a way that will allow maintenance workers to receive an alert of the trash cans that need to be collected, so that they can plan an effective route.

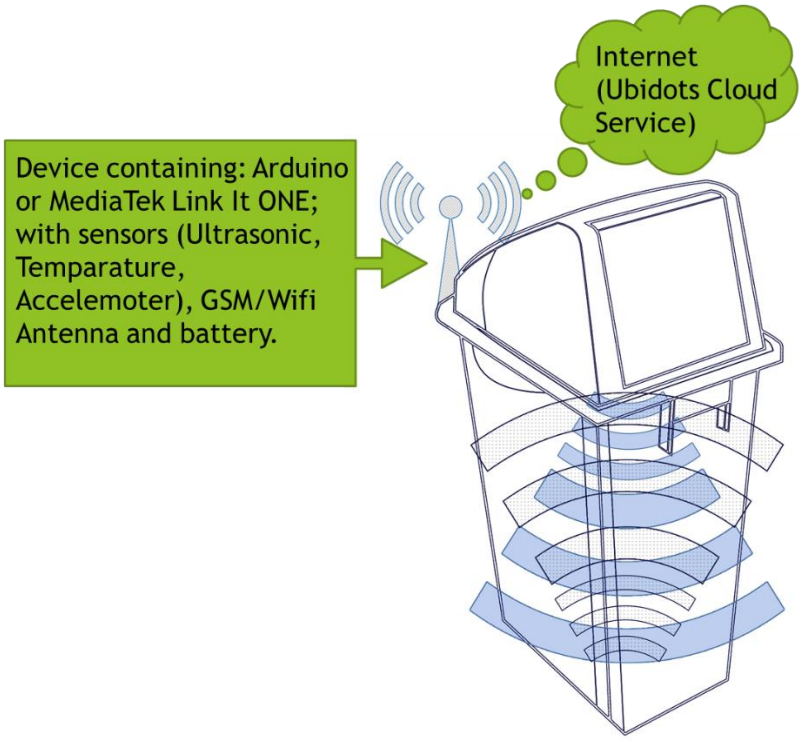



*Fig 1-3: The Eco-Smart Can concept.*

# 2. Lab Materials, Circuits, Programs

For this project I tested and used multiple materials, some did work and some did not. In this session I am listing the materials and circuits that I needed to help me accomplish this project.

## 2.1 Materials

- MediaTek Labs LinkIt ONE Development Board
- GSM Antenna
- WIFI Antenna
- HC-SR04 Ultrasonic sensor
- Groove Temperature & Humidity sensor (HDC1000)
- Groove 3 Axis Digital Accelerometer (±16g)
- Groove Base Shield
- Groove Universal four pin buckled cable
- Male to Female 2.54 mm Dupont Jumper wires
- Basic micro-USB Cable
- Polymer Li-ion 1050 mAh battery
- Ting 2G SIM Card
- Wireless Router

## 2.2 Software & online service

- Arduino IDE
- Ting 2G account
- Ubidots cloud service

# 3. MediaTek LinkIt ONE basic circuitry

According to the MediaTek INC website, MediaTek LinkIt™ ONE development platform enables you to design and prototype Wearables and Internet of Things (IoT) devices, using hardware and an Application Programming Interface (API) that are similar to those offered for Arduino boards (What is MediaTek LinkIt ONE development platforms?, .n.d.). The platform is based around the world's smallest commercial System-on-Chip (SOC) for Wearables, MediaTek MT2502 (Aster). This SOC chip features:

- CPU core: ARM7 EJ-S 260MHz
- Memory: 4MB RAM, 4MB Flash
- Dual Bluetooth 2.1 (SPP) and 4.0 (GATT)
- GSM and GPRS modem

While the development platform also has Wi-Fi, GPS, Audio codec, and SD card connector (What is MediaTek LinkIt ONE development platforms?, .n.d.). Check the Appendix A for further circuit details.

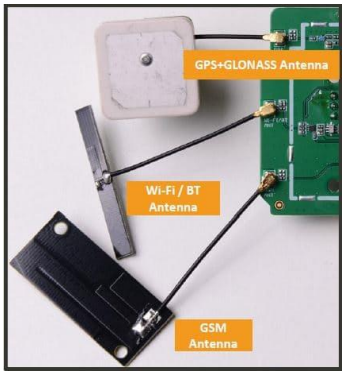



*Fig. 3-1: Antenna Ports (MediaTek Labs, 2015)*

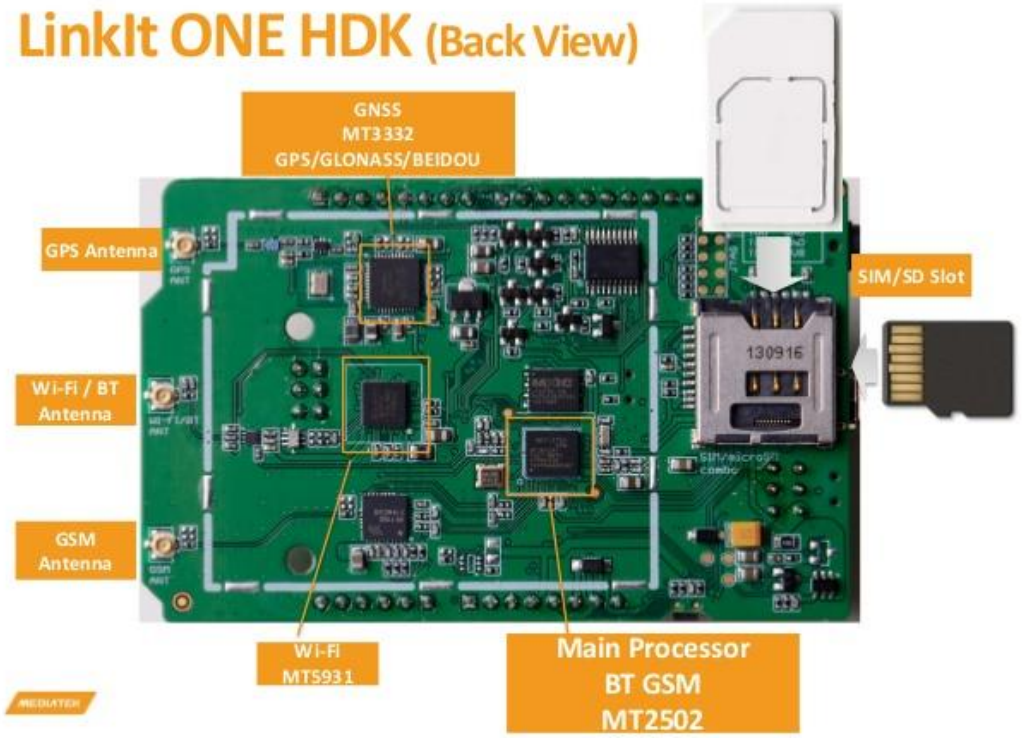



*Fig. 3-2: Back view, chips names (MediaTek Labs, 2015)*


*Fig. 3-3 front view: different pins and inputs (MediaTek Labs, 2015)*

## 4. Circuit Layout

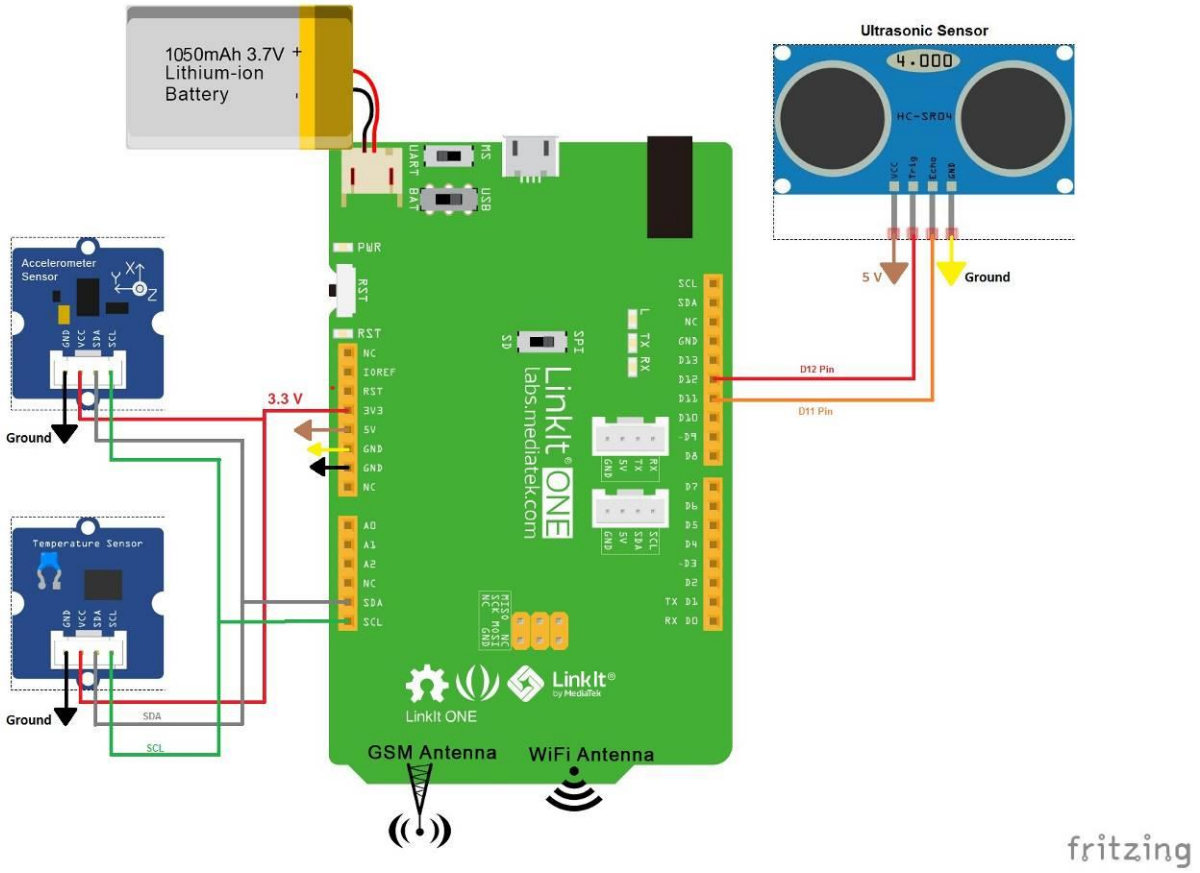



*Fig. 4-1: Board connection layout (Developed with Fritzing Software)*

## 5. Circuit and components connection

For our device to be able to send data from the sensors we have to be able to connect our development board to the internet. The GSM and Wi-Fi antenna is connected to GSM and Wi-Fi antenna port on our LinkIt ONE development board. The GSM antenna is provided in the LinkIt ONE development board's kit and supports 2G GSM network standards. While the Wi-Fi antenna only supports 802.11 b/g/n protocol. The Grove base shield is connected on the top of LinkIt ONE as seen in Fig. 5-1. Seed Studio states that the Groove base shield is an expansion board, that has many Grove connectors, making it convenient to use Grove products together as

shown in Fig. 5-2. Grove is a modulated, ready-to use tool set. Much like Lego, it takes a building block approach to assembling electronics. The groove system consists of a base shield and various modules with standardized connectors (Seed Studio, 2012).

*Fig. 5-1: GPRS & Wi-Fi antenna & Ting 2G Sim card.*

The Grove Humidity and Temperature sensor HDC1000 is connected to the I2C Serial Bus Address Configuration grove connector of the Grove base shield i.e. to voltage (3.3V), ground (GND), SCL and SDA pins of LinkIt ONE (Fig. 5-2). HDC1000 sensor which was designed by Texas Instruments measures temperature and humidity based on a novel capacitive sensor. The Grove websites confirms that the humidity and temperature sensors are factory calibrated (Grove - Temperature&Humidity Sensor (HDC1000), n.d.). The detecting range of HDC1000 sensor for the humidity is 0% RH to approximately 100% RH, and for the temperature is -40°C to approximately 125°C. Its accuracy reaches up to ±3% RH and ±0.2°C. It is a fairly accurate sensor and the readings consume low power. The humidity based on a novel capacitive sensor. The humidity and

temperature sensors are factory calibrated (Grove - Temperature&Humidity Sensor (HDC1000), n.d.).

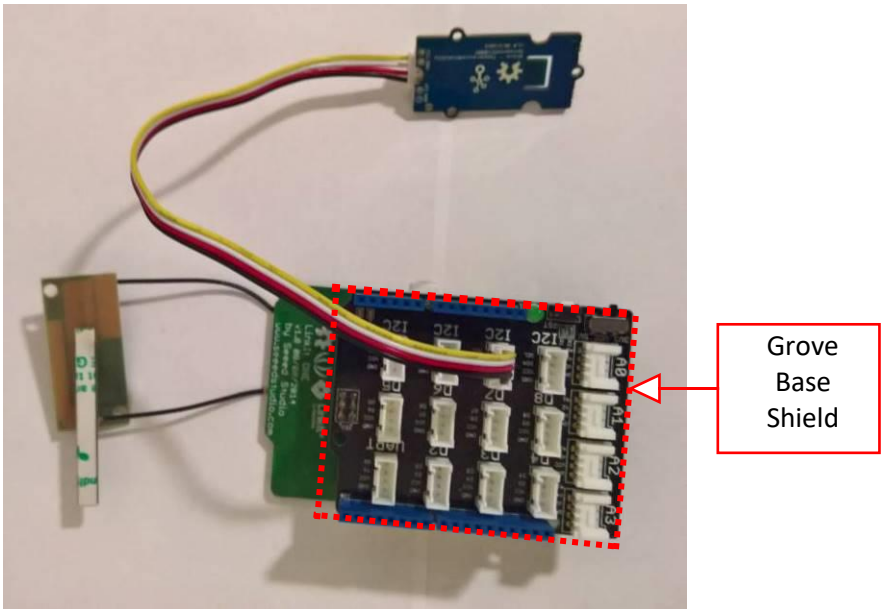



*Fig. 5-2: Temperature & Humidity sensor*

Grove - 3-Axis Digital Accelerometer (±16g) sensor is connected to another I2C grove connector of the Grove base shield i.e. to voltage (3.3V), ground (GND), SCL and SDA pins of LinkIt ONE board as seen in Fig. 5-3.

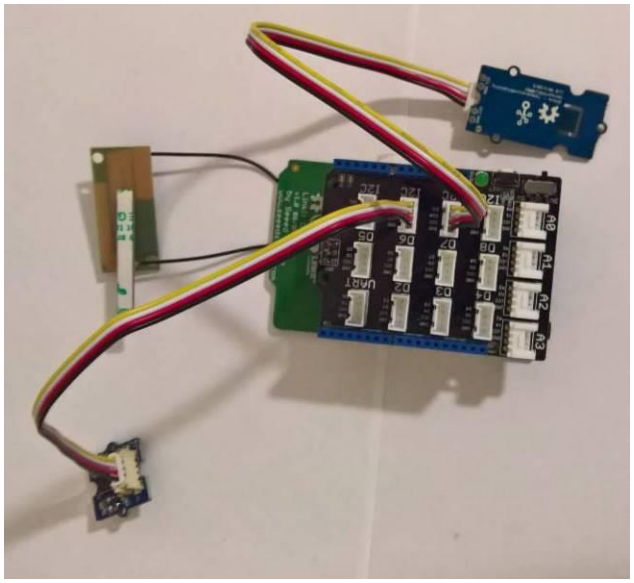


*Fig. 5-3: Accelerometer sensor*

The HC-SR04 Ultrasonic sensor Vcc (Brown wire) is connected to the 5V of Grove Base shield i.e. to 5V of LinkIt ONE board. The ground Gnd (Yellow wire) to the ground Gnd of Grove base shield i.e. to Gnd of LinkIt ONE board. The Trigger pin (Red wire) to D12 pin of Grove base shield i.e. of LinkIt ONE board. The Echo pin (Orange wire) to D11 pin of Grove base shield i.e. of LinkIt ONE board as shown in Fig. 5-4.

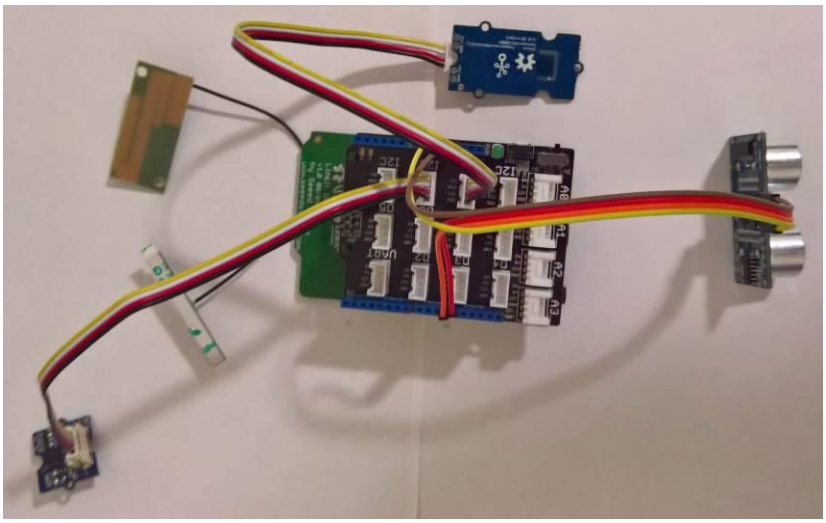


*Fig. 5-4: Ultrasonic sensor (HC-SR04)*

At each sensor connection, I used online codes or custom codes in the libraries of the LinkIt ONE SDK in Arduino IDE to test the sensors and make sure they work individually. Once all connections were successful, I connected the 1050mAh Li-Po battery contained in the LinkIt ONE development kit to the LinkIt ONE board. (See Fig. 5-5)

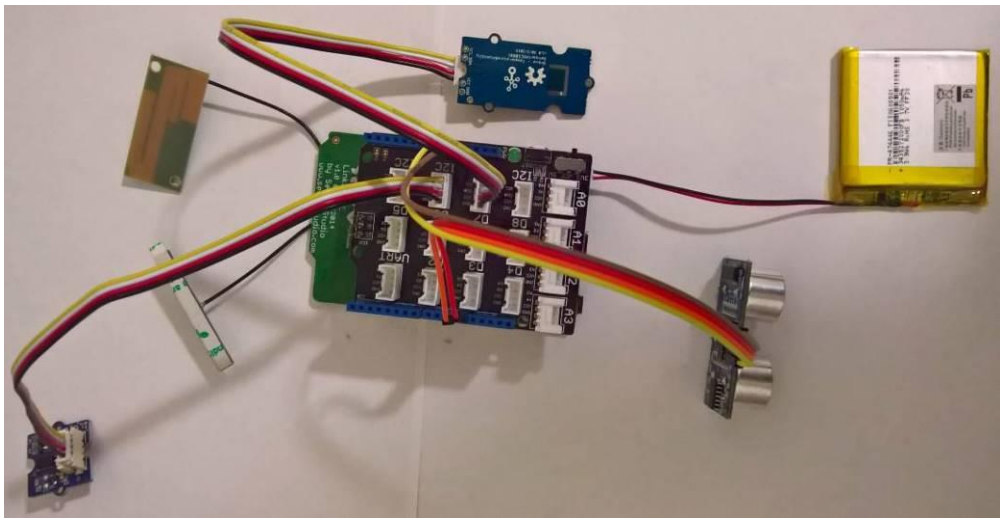


*Fig. 5-5: Li-Po battery added*

## 6. Ubidots cloud service

ProgrammableWeb reveals that Ubidots offers a platform for developers that enables them to easily capture sensor data and turn it into useful information (ProgrammableWeb, 2013). I used the Ubidots cloud service to collect the data sent through internet from the Eco-Smart can device. The next diagram illustrates how my prototype board will work in conjunction with the Ubidots cloud service.

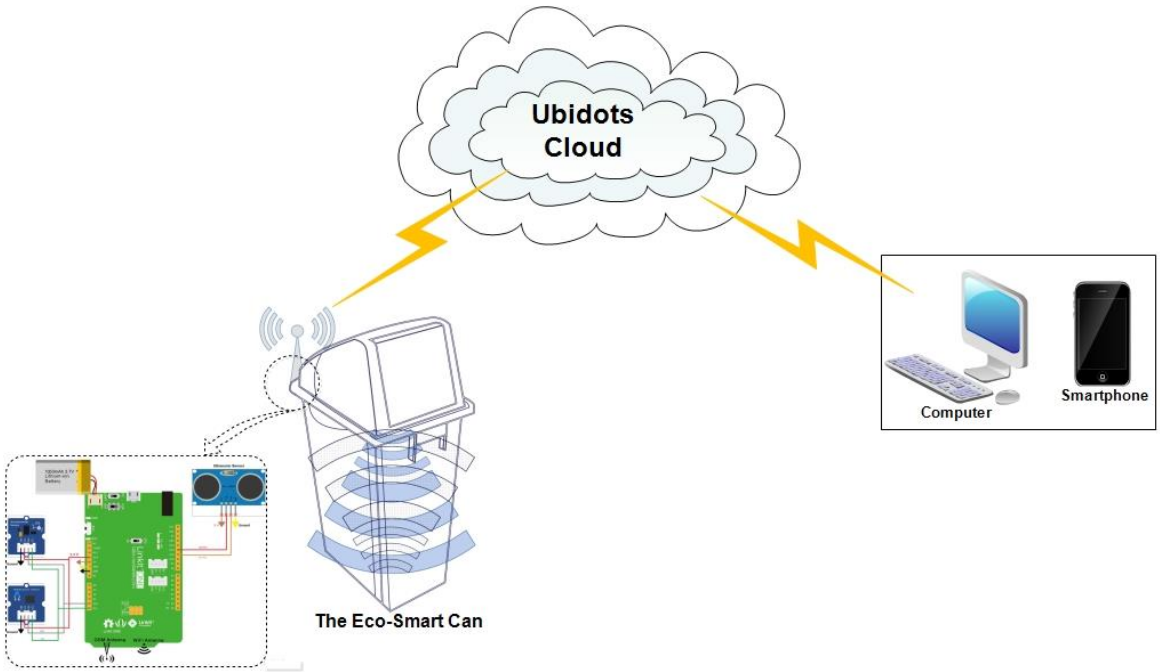



*Fig. 6-1 Ubidots cloud service concept (Developed with Edraw Max Software)*

There are many cloud services such as Amazon Web Service Iot (AWS IoT), ThingSpeak, Google Cloud, or MCS etc.; but after few searches, I discovered that the Ubidots website offers a free plan that had more features needed for this project than the others. The Ubidots platform free plan allowed me to send data to the cloud from any Internet-enabled device. The service, as pointed out on the Ubidots' website is also very affordable with up to 500,000 data point uploads (or dots) a month, with 1 Month of historical storage, 1,000 email alerts and 5 SMS alerts for free (Ubidots, n.d.). In addition, it is really user friendly, have multiple forums and the Application Programming Interface (API) is well documented. After, registration and activation of my account on Ubidots account; I created different variables (trash level, temperature, battery level, security) and connected each to an appropriate display widget, (as seen in Fig. 6-1). Finally, for this project I configure actions and alerts based on the real-time data from the sensor of the trash can. The two alerts will automatically send emails. The first one is when the Trash Can Level reaches level 95% (different trash level percentage can be used) an email alert will be sent to the Maintenance Facility saying that the trash is full and needs attention (See Fig. 9-6). Additionally, I can set rules that will trigger it to send an SMS to a defined phone number. The second rule is for security reasons; it will send an email when someone is tempering with the trash can or when it suddenly changes location (See Fig. 9-9). Each variable ID created was used in my coding in order to send the right data to the right widgets.

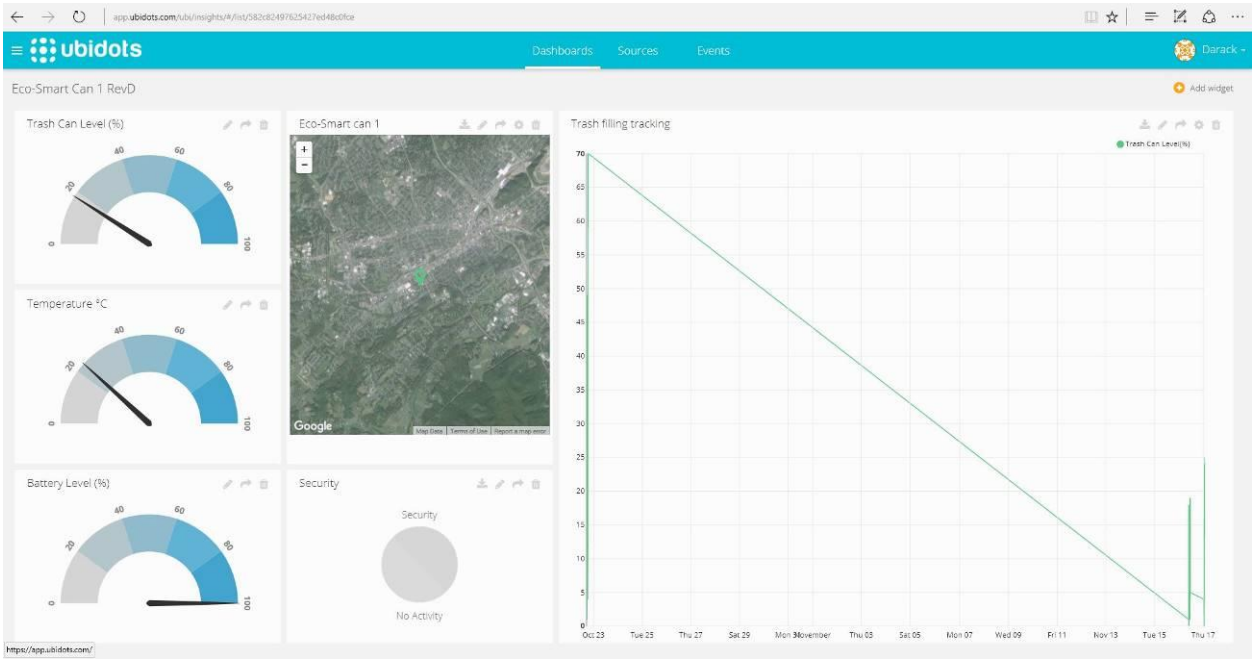



*Fig. 6-1 Ubidots Dashboard for the Eco-Smart Can*

# 7. Coding

Below is the final code for the prototype of the Eco-Smart Can. It was developed in Arduino IDE using C/C++ programming language. In this code also knows Arduino Sketch, I use two ways (Wi-Fi and GPRS) for my board to connect to the internet in order to send data to the cloud service. I did that because I had a hard time connecting to the internet using GPRS; after few research and testing I found that the problem was coming from the coverage of the company from whom I purchased the SIM card, therefore I decided to use Wi-Fi as the default connection instead of GPRS. I had trouble with the Wi-Fi connection too, because our campus Wi-Fi security type encryption (i.e. WPA2-Enterprise) is not supported by the MediaTek LinkIt ONE board. To overcome the issue, I used a wireless router to create a custom access point that has a supported security type by the LinkIt ONE board. After Ubidots setup and Arduino code completed, the code was verified and compiled in the Arduino software. I connected the LinkIt ONE board through the Basic micro-USB cable and the codes were uploaded to the board. Below is the code running in

the board. I added comments to make it easy to understand what each line does. Check Appendix B for the full code.

# 8. Overview of the Arduino Sketch

## 8.1 General Codes

MediaTek labs, states that an Arduino sketch "is a source code file representing the core controlling logic for the LinkIt ONE development board. It consists of two main structures: setup and loop" (MediaTek Labs, n.d.).

In this section I will discuss what the above code do. First, LinkIt ONE initializes the Ubidots account, then the sensors, Wi-Fi or GPRS, the location and finally the serial port. To minimize battery consumption, I did not use the GPS antenna to detect and acquire automatically the location. I manually added the location of the device in the codes using latitude and longitude coordinates. The Eco-Smart Can device will try to update the collected data to Internet on the Ubidots cloud service after each 30 seconds interval which can be changed depending on the location of each Eco-Smart Can requirements (i.e. the busier the location the shorter the reading interval). Each 30 seconds the Eco-Smart Can device will read and update the Ubidots cloud service with any changes in the data from the ultrasonic, accelerometer and temperature & humidity sensors, plus the battery level.

## 8.2 Ultrasonic sensor overview

This part will explain the basic principle of how the ultrasonic sensor works in the Eco-Smart can prototype. Cytron Technologies explain in their user's manual of the HC-SR04 sensors that the ultrasonic sensor uses sonar to determine distance to an object like bats or dolphins do (Cytron Technologies, 2013). To start the measurement, the code triggers the digital Trig pin (Pin D12) of the sensor with a high (5V) level signal of 10µs (micro second), this initiate the sensor to transmit out eight cycles of ultrasonic burst at 40kHz (kilohertz) and wait for the reflected ultrasonic burst. When the signal comes back and is detected by the sensor; the sensor will set the Echo pin (pin D11) to high (5V) and delay for a period (time) which proportion to distance taken by sonar waves to travel from the sensor to the object and back to the sensor. To obtain the distance, we need to first divide time by 2. We know that, *Distance = Speed × Time*; in this case, *Trash distance = (Echo pin high time/2) × velocity of sound (340m/s).* The range of the HC-SR04 sensor is from 2 cm - 400 cm with an accuracy of 3 mm and the Velocity of sound in dry air at 20 °C (68 °F) is 340 m/s. Since the duration for which the trig pin and echo pin stays high is in microseconds (µs). Therefore, *Velocity of sound in cm/µs = (340m/s × 100)/1000000 = 1/29.1 cm/µs*. Finally, *Trash distance (cm) = (Echo pin high time in µs/2) x (1/29.1 cm/µs) = Echo pin high time/58.2 cm.* (Cytron Technologies, 2013). See data sheet in Appendix *B.*

# 9. Theory and Results

Once the LinkIt ONE board in the Eco-Smart Can is turned on and successfully connect to the internet. It initializes the Ubidots account, then the sensors, Wi-Fi or GPRS, the location and finally the serial port. To get consistent and reproducible data for the test of the device, I created a simple system that will reproduce how a traditional trash work as seen in Fig. 9-1. In this system,

as the blue box that represent the trash move forward toward the ultrasonic sensor the trash level increases. A ruler in cm was tape on the side and an indicator was taped on my box to check on the accuracy of the device when measuring.

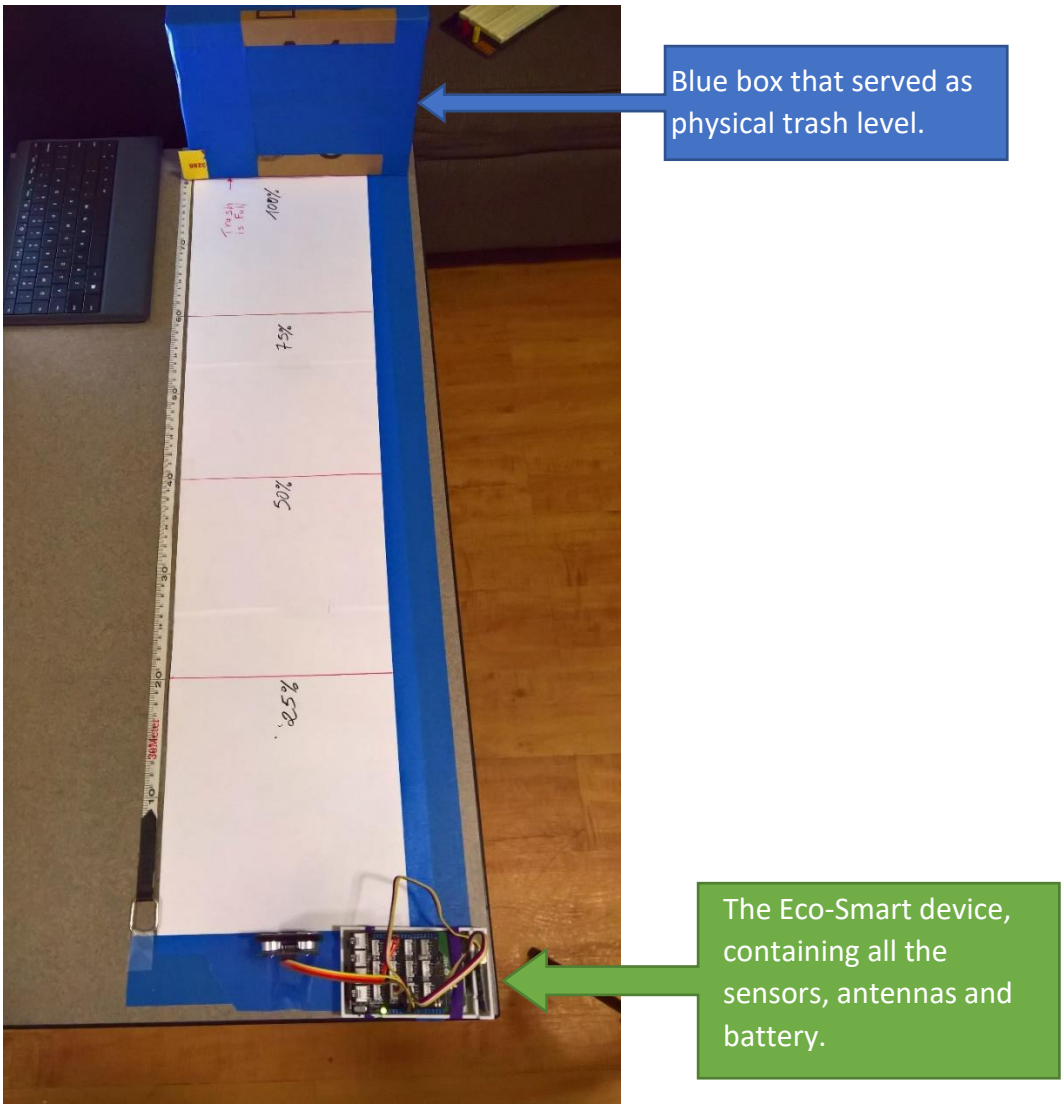



*Fig. 9-1: Device testing system.*

LinkIT ONE Board has one serial port on which the micro-USB cable is connected for the serial monitor. The serial monitor contained in the Arduino IDE prints the data from the sensors before it is uploaded on the cloud platform. See Fig. 9-2.

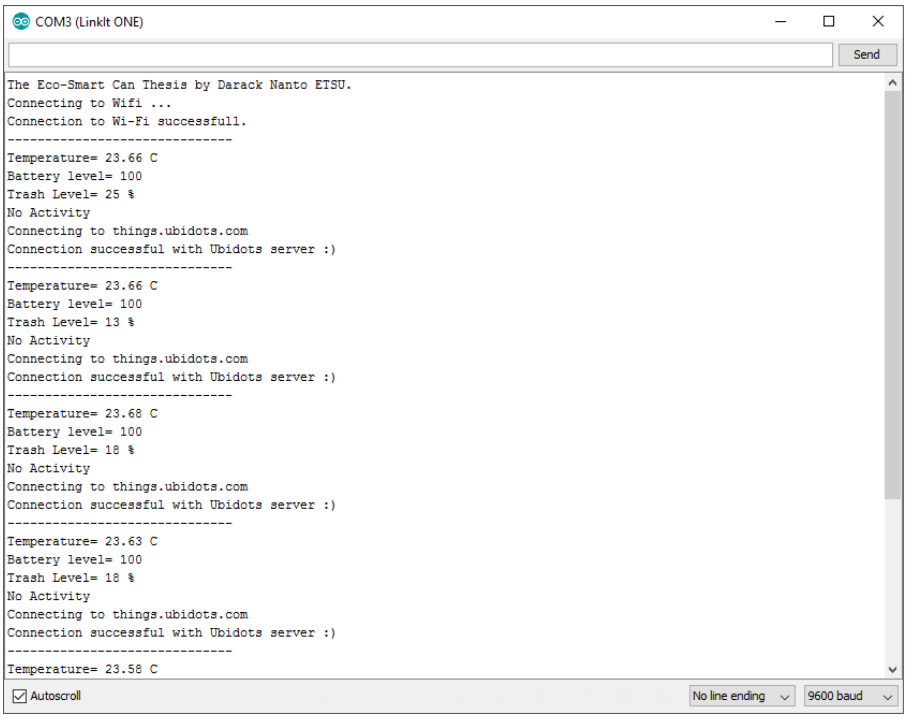



*Fig. 9-2: Arduino IDE Serial Monitor*

The Eco-Smart can update the data to the Ubidots platform after every 30 seconds intervals. The Ubidots platform collects the data sent from the 3 sensors plus the battery level and converts it to meaningful outputs through different widgets, as seen in Fig. 9-3.

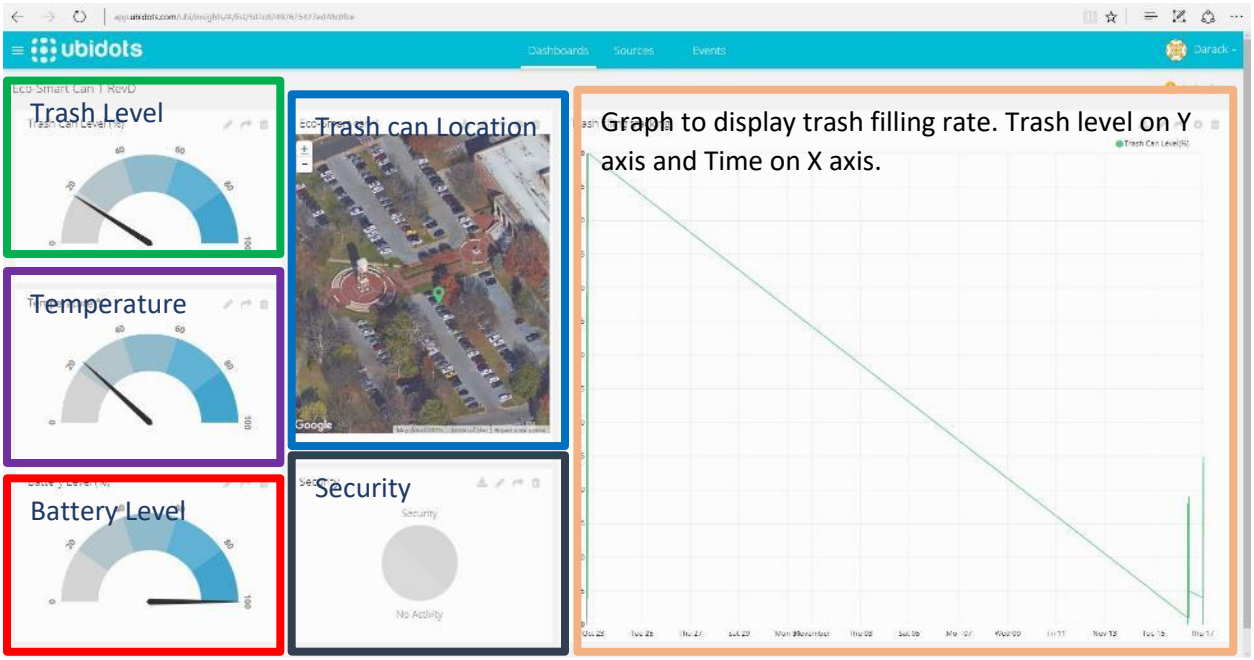



*Fig. 9-3: Ubidots cloud service Dashboard*

Trash level in the Eco-Smart Can is measured in percentage (0%- Empty and 100%-Full). Temperature sensor reading is in Celsius. Security detects if any unusual activity is taking place. If someone tries to move the device, it reacts. Battery level is in percentage, LinkIt ONE board only return 4 possible values for the battery level: 100%, 66%, 33% or 0%.

Once the trash is full (100%), the gauge in Ubidots dashboard indicates that the trash is full and it will trigger an email to be sent since the trash level exceeded 95%. See the below figures.

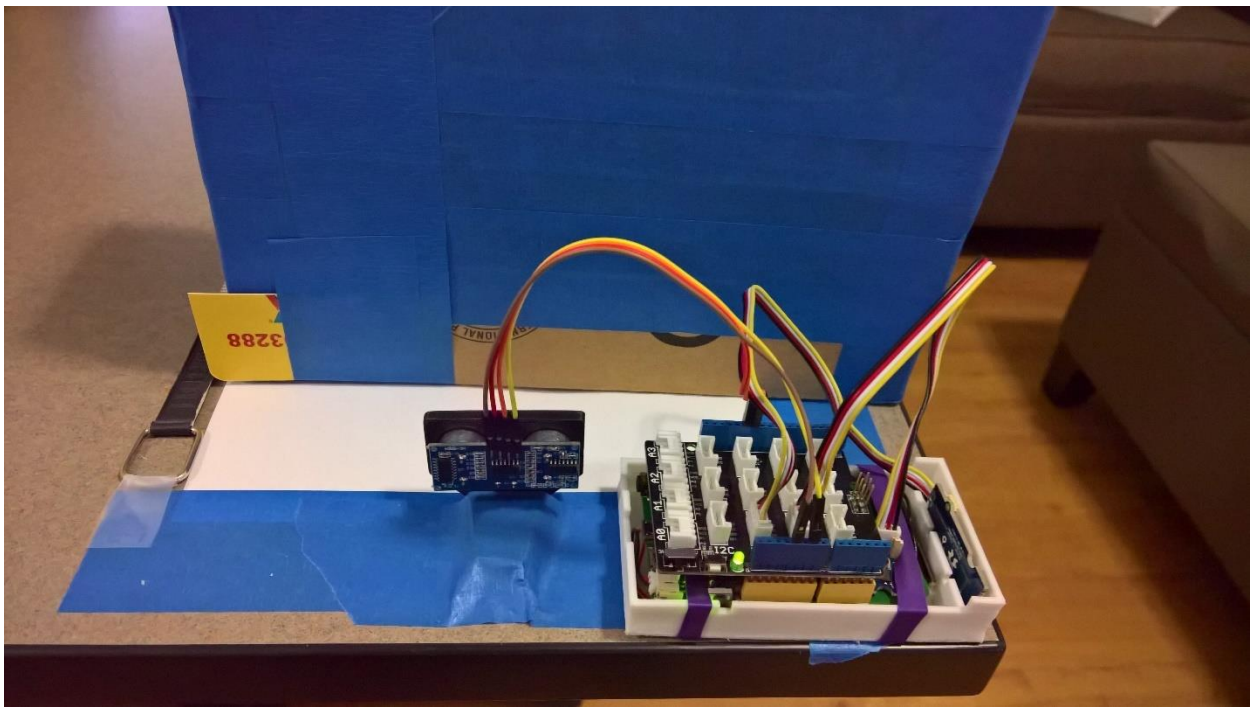


*Fig. 9-4: Device Test - Trash full (100%)*

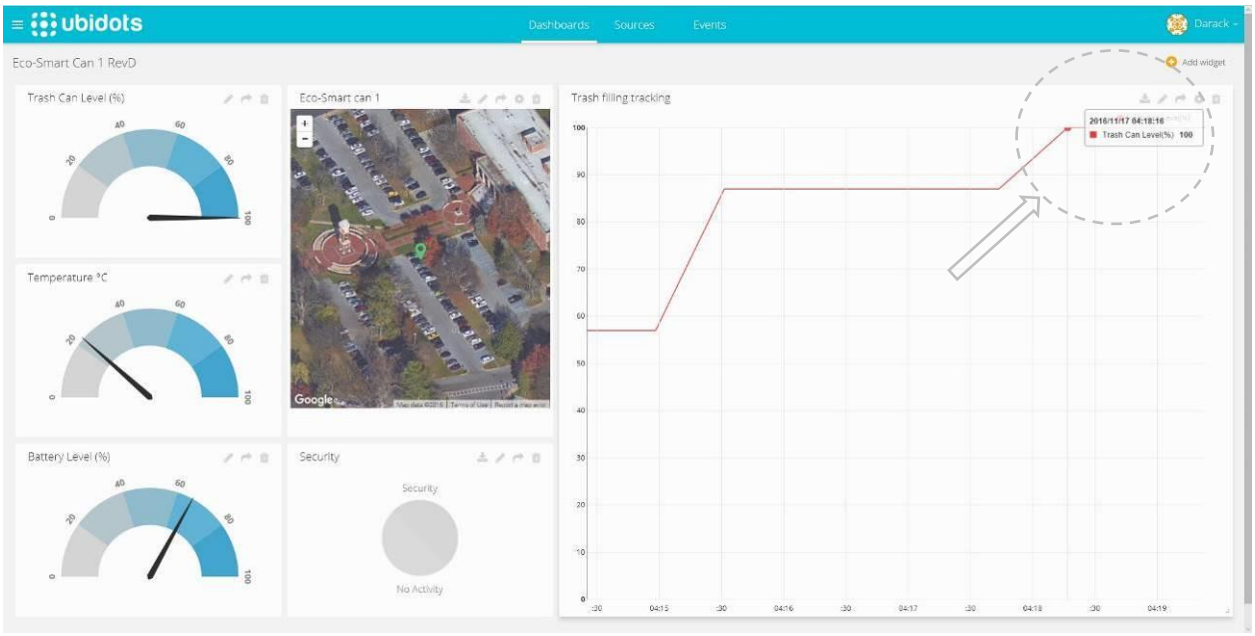



*Fig. 9-5: Ubidots dashboard- Trash full (100%)*

Email sent to Maintenance facility when the trash got full.

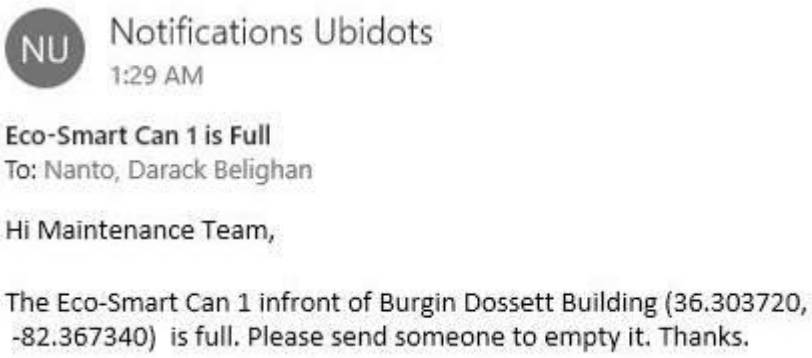



*Fig. 9-6: Email alert for trash full*

As I try to move the Eco-Smart Can device from right to left (simulating a vandalism scenario), it sends an alert to the Ubidots account, then send an email to Maintenance Facility or Public Safety that the trash is being tempered (see Fig. 9-7). When unauthorized activity occurs the indicator widget in the dashboard turns green and the email is sent (see Fig. 9-8 and 9-9).

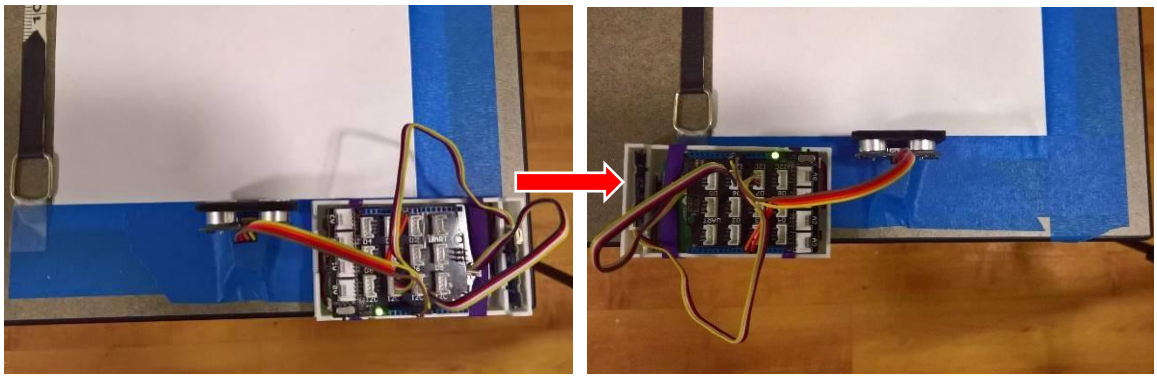


*Fig. 9-7: Moved device from right to left, to simulate a change in position.*


*Fig. 9-8: Widget turns grey (No activity) to green (Unusual Activity) when devices changes position.*

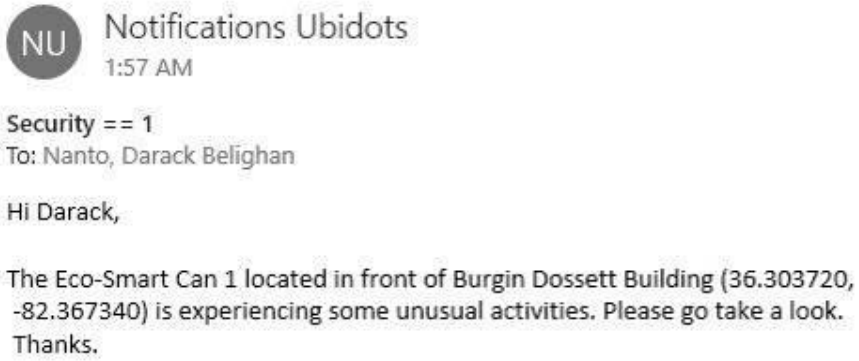



*Fig. 9-9: Email alert for unauthorized activities*

The temperature & humidity sensor works great too, since the testing was done in a control environment the temperature was constant with minimal changes. This is why the temperature values is almost similar in all my testing, and no obvious changes were noticed. See Appendix A for The temperature & humidity sensor (HC1000) detailed circuitry.

I have done my best to show in this project that IoT can improve the work of our campus maintenance team and reduce cost/resources and disease. This project seemed complicated at the beginning but through this documentation I was able to show how it can be done.

## 10. Instructional Capture

This project was a really long project and it took me many trials and errors to make it work. Through the journey of prototyping the Eco-Smart Can I learned countless things. The first thing I learned is how microcontrollers and microcontroller platforms works. I learned more about the Arduino and MediaTek LinkIt ONE platforms. Since LinkIt ONE board uses a simplified version of C/C++ programming language; I learned how to write code in C/C++ and compile it to a development board. In addition, I learned how the Internet of Things (IoT) works and its countless application. The main component of this project was the ultrasonic sensor, I learned how to use the ultrasonic sensor and make calculations to display an accurate reading of the distance of an object. Making all my sensors work in harmony was the key of this project, I learned how to interface many sensors at once. Using the Ubidots cloud service, I learned how data was sent to the internet using different internet protocol. Finally, I learnt all the stages of prototyping and apply them.

Through this project, I had many issues. Most of them were simple issues such as connecting the wrong cable, missing semi column in coding, and also problems with defective sensors. These problems were fixed by troubleshooting my circuit and reading the error log of codes. A major problem I had was the connection of my board to the internet through GPRS, after reading on forums and going on the website of the provider I have seen that the coverage

for Johnson City was really bad. Once I finished creating the prototype, I could not find a way to place the ultrasonic sensor and the accelerometer sensor, so I just design a custom 3D case to hold them in place as seen in Fig. 9-4. See Appendix C for the 3D previews.

## 11. Future improvement

Internet of Things is a process of continuous study and learning it never end. This project can be taken further. In the future, I will like to improve this device by adding a solar panel to charge the battery to make it autonomous, with minimum human intervention. In addition, create a real-time monitoring of the civic body's garbage vehicles using Radio Frequency Identification (RFID). Each maintenance staff has to scan his/her RFID card so that who and when and at what time trash cans were emptied is recorded. It will help ensure that everyone is doing their job. The velocity of the sound that I am using in my code, is when the sound travels in a dry air at 20 °C (68 °F), but since the trash will be exposed to different weather sometimes our readings will be affected. Therefore, I think to fix this issue we can use the data of the temperature & humidity sensor to compensate the changes in order to keep our reading accurate. This project has countless way to improve it.

## 12. References


Arduino. (n.d.). *WHAT IS ARDUINO?* Retrieved May 20, 2016, from https://www.arduino.cc/

Au, J., & Gertz, E. (2016). *3D CAD with Autodesk 123D : Designing for 3D printing, laser cutting, and personal fabrication* (First ed., Safari Tech Books).

Behmann, F., & Wu, K. (2015). Collaborative internet of things (C-IoT) : For future smart connected life and business (Safari Tech Books).

Cytron Technologies. (2013, May). *Product User's Manual – HCSR04 Ultrasonic Sensor* [PDF].

EdrawSoft. (2013, November 1). *Edraw Max* (Version 7.2) [Computer software]. Vers. 7.2, 1 November 2013. Retrieved November 10, 2016, from https://www.edrawsoft.com/edraw-max.php

Escobar, R., & Perez-Herrera, C. (2015). Low-cost USB interface for operant research using Arduino and Visual Basic. *103*(2), 427-435.

Fang Zhanzhao. (n.d.). *Grove - Temperature&Humidity Sensor(HDC1000) v1.0 sch* [PDF].

Friends-of-Fritzing Foundation. (2 June 2016). *Fritzing* (Version 0.9.3b) [Computer software]. Retrieved October 5, 2016, from http://fritzing.org/download/

Johnson, J., & East Tennessee State University. Dept. of Technology. (2005). *Identifying Common Ultrasonic Predictive Failure Signatures in Bearing Elements for the Development of an Automated Condition Based Ultrasonic Monitoring Controller*.

MediaTek Labs. (2015, July 12). *MediaTek Labs Webinar: Getting Started with LinkIt* ONE [digital image]. Retrieved November 9, 2016, from http://www.slideshare.net/MediaTekLabs/mediatek-labs-webinar-getting-started-with-linkit-one

MediaTek Labs. (n.d.). *Weather Station Tutorial*. Retrieved July 23, 2016, from https://labs.mediatek.com/site/global/developer_tools/mediatek_linkit/documentation/weather-station/tutorial/

Mohammed Shahanas, & Bagavathi Sivakumar. (2016). Framework for a Smart Water Management System in the Context of Smart City Initiatives in India. *Procedia Computer Science, 92*, 142-147.

O'Neill, T., & Williams, J. (2014). *Arduino*.

ProgrammableWeb. (2013, November 25). *Ubidots*. Retrieved November 10, 2016, from http://www.programmableweb.com/api/ubidots

Sachse, H. (1975). *Semiconducting Temperature Sensors and Their Applications*.

Seed Studio. (n.d.). *Base Shield V2*. Retrieved June 22, 2016, from https://www.seeedstudio.com/Base-Shield-V2-p-1378.html

Seed Studio. (n.d.). *Grove - Temperature&Humidity Sensor (HDC1000)*. Retrieved August 10, 2016, from https://www.seeedstudio.com/Grove-Temperature%26Humidity-Sensor-(HDC1000)-p-2535.html

Seed Studio. (2012, February 28). *Introduction to Grove* [PDF].

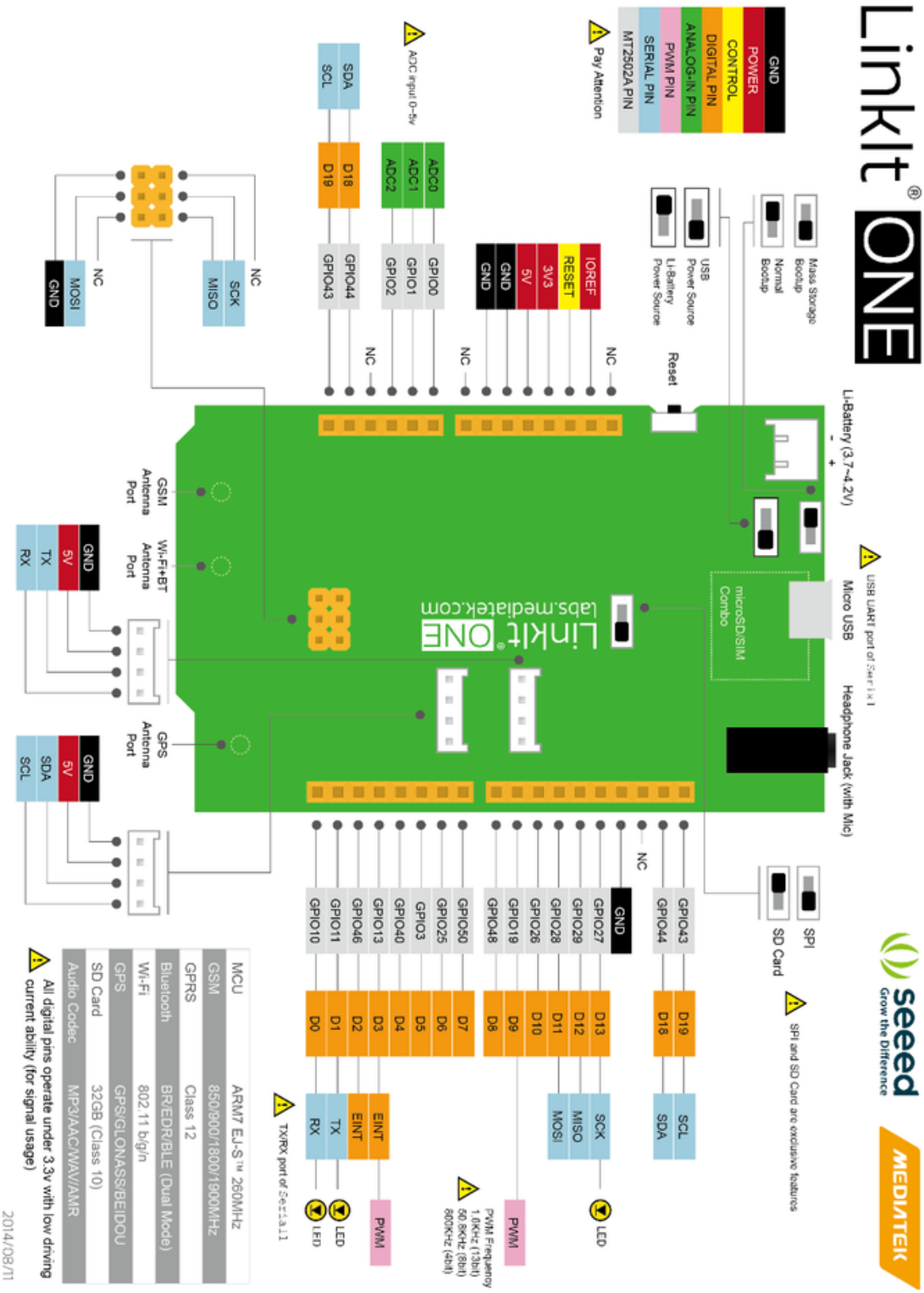
Scuotto, V., Ferraris, A., & Bresciani, S. (2016). Internet of Things. *Business Process Management Journal,22*(2), 357-367.

Ubidots. (n.d.). *Flexible pricing to take your project from prototype to production*. Retrieved July 20, 2016, from https://ubidots.com/pricing

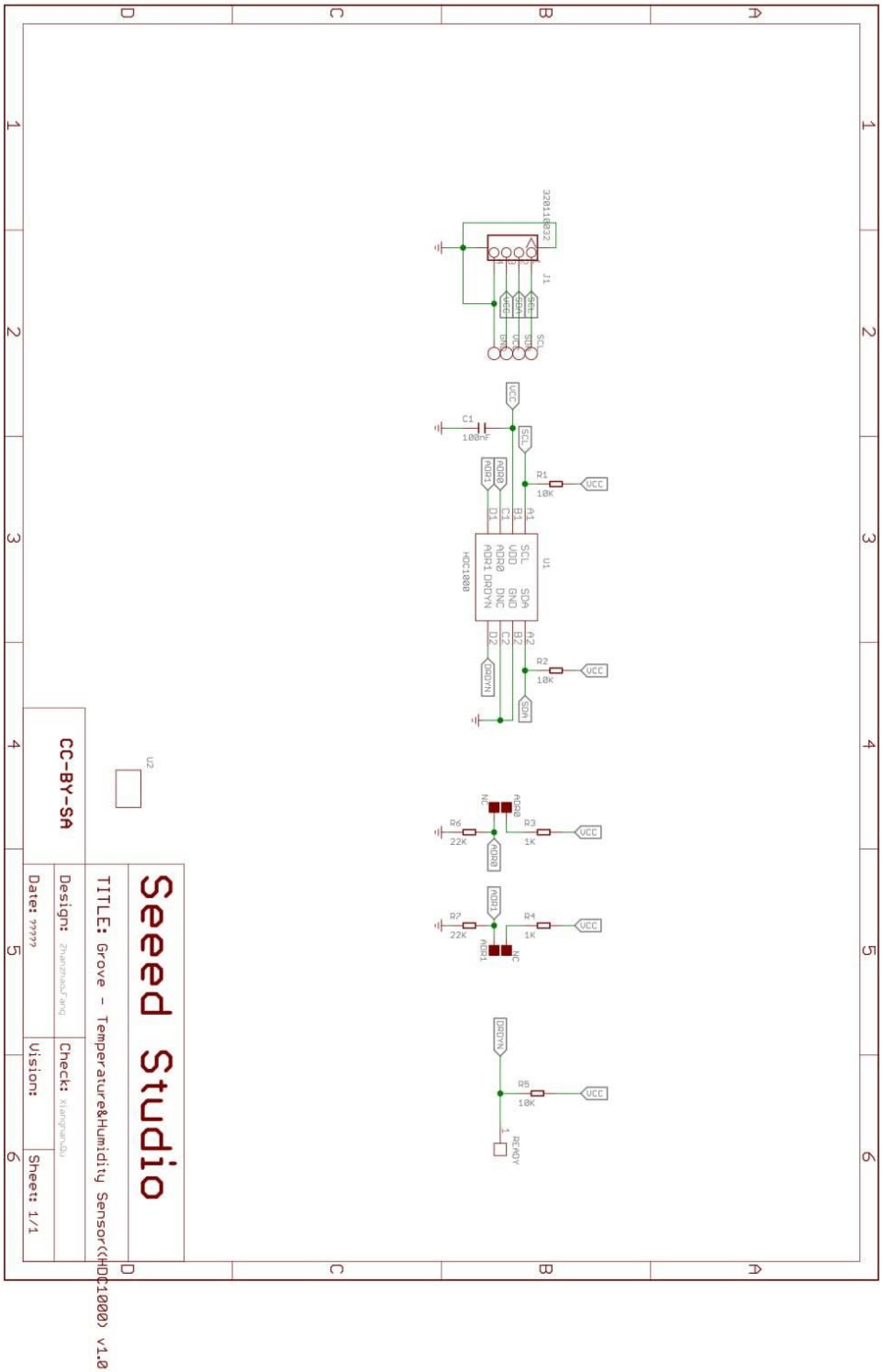
*What is MediaTek LinkIt ONE development platforms?*. (n.d.). Retrieved July 7, 2016, from https://labs.mediatek.com/site/global/developer_tools/mediatek_linkit/whatis_linkit_one/index.gsp

Yuen, K., Woo, P., Ip, M., Liang, R., Chiu, E., Siau, H., . . . Chan, T. (1997). Stage-specific manifestation of mold infections in bone marrow transplant recipients: Risk factors and clinical significance of positive concentrated smears. *Clinical Infectious Diseases, 25*(1), 37-42.

# Appendix A




MeidaTek LinkIT ONE Board




Grove - Temperature Humidity Sensor (HDC1000)

# Appendix B

```
/*
  - Project: The Eco-Smart Can
  - Author: Darack Nanto
  - Mentor: Dr. Paul Sims
  - Description: Making trash collection system efficient on campus.
              Thus, helping in optimization of route for trash
              collection; saving fuel.
  -Created: July 19, 2016
  -Last modified: November 15, 2016
  -Version: RevE

  Copyright (c) 2014 MediaTek Inc.  All right reserved.

  This library is free software; you can redistribute it and/or
  modify it under the terms of the GNU Lesser General Public
  License as published by the Free Software Foundation; either
  version 2.1 of the License.

  This library is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
  See the GNU Lesser General Public License for more details.

  HDC 1000 Library can be downloaded from:
  https://http://wiki.seeed.cc/Grove-TemperatureAndHumidity_Sensor-HDC1000/

  Accelerometer ADXL345 Library can be downloaded from:
  http://www.seeedstudio.com/wiki/File:DigitalAccelerometer_ADXL345.zip
*/


#include <LBattery.h>
#include <LGPRS.h>
#include <LGPRSClient.h>
#include <LWiFi.h>
#include <LWiFiClient.h>
#include <Wire.h>
#include "Adafruit_HDC1000.h"
#include <ADXL345.h>

//Wi-Fi
#define WIFI_AP "EcoSmartCan"
#define WIFI_PASSWORD ""
#define WIFI_AUTH LWIFI_OPEN  // choose from LWIFI_OPEN, LWIFI_WPA, or
LWIFI_WEP according to your WiFi AP configuration

//Ultrasonic Module
#define ECHOPIN 11        // Digital Pin 11 to receive echo pulse (Orange
Wire)
#define TRIGPIN 12        // Digital Pin 12 to send trigger pulse ((Red Wire)

//Ubidots account data
#define URL    "things.ubidots.com"
#define TOKEN  "nnVDZfejcBSLiCASBj23oU1bw4U25l"     // replace with your
Ubidots token generated in your profile tab
#define VARID1 "5806519e7625426c268d5a68"    //Trash level ID variable in
Ubidots
```

```
#define VARID2 "5806508776254265e7d2be8f"   //Temperature of trash can ID
variable in Ubidots
#define VARID3 "580651dd7625426c98318b03"  //Activity tracking ID variable in
Ubidots
#define VARID4 "580651eb7625426d4069ca14"  //Battery level tracking ID
variable in Ubidots

// Reading temperature or humidity takes about 250 milliseconds!
Adafruit_HDC1000 hdc = Adafruit_HDC1000(); //Temperature and Humidity sensor
readings from HDC100 sensor

//Accelerometer sensor connected to I2C of LinkIT One
ADXL345 accel; //variable accel is an instance of the accel345 library

//NB: LinkIt ONE Board only return 4 possible values for the battery level:
100%, 66%, 33% or 0%.
char buff[256];  //Info about battery level and charging

long duration, distance, Level;

//ETSU Location where the prototype is placed!! (I choose the
String loclat="36.303720";
String loclng="-82.367340";
String Location;

unsigned long ReportingInterval = 30000;  // How often do I want to update
the IoT site in milliseconds (in this case 30 seconds)
unsigned long LastReport = 0;             // When was the last time you
reported

//Accelerometer adxl345 outputs
int x_initial,y_initial,z_initial;
double xyz_initial[3];
double ax_initial,ay_initial,az_initial;

/*LGPRSClient client;  //GPRS Client*/ //Use this line when 2G sim card is
inserted

LWiFiClient client;  //Wifi Client

void setup()
{
  Serial.begin(9600);    //For serial debugging on Laptop/computer
  Serial.println("The Eco-Smart Can Thesis by Darack Nanto ETSU.");
  accel.powerOn();       //Powering Accelerometer
  hdc.begin();             //Initializing HC1000 Temperature / Humidity
sensor

  //Initializing Ultrasonic sensor pins
  pinMode(ECHOPIN, INPUT);
  pinMode(TRIGPIN, OUTPUT);

  axis_initialize(); //Initializing Accelerometer

  //Location String containing lat and long data as required by Ubidots API
  //For more info:
http://ubidots.com/docs/get_started/quickstart/tutorial_geopoint.html
```

```
  Location="{\"lat\":";
  Location=Location+loclat;
  Location += " ,\"lng\":";
  Location=Location+loclng+ "}";

  //Serial.println("Attach to GPRS network by APN setting");      // For  2G
Sim card
  //while (!LGPRS.attachGPRS("wholesale","UserName","Password")) //Enter
correct Access Point Name (APN) according to your GSM provider. Username and
password if not present then keep it empty.

  Serial.println("Connecting to Wifi ...");
  while (0 == LWiFi.connect(WIFI_AP, LWiFiLoginInfo(WIFI_AUTH,
WIFI_PASSWORD))) //Enter correct Access Point Name (APN) in the variable up.
  {
    delay(500);
    Serial.println("The Eco-Smart Can Wi-Fi connection in progress");
  }

  //LGPRSClient globalclient ; //Client has to be initiated after GPRS is
established with the correct APN settings. USe for 2G Sim Card
  //client = globalclient;     //Again this is a temporary solution described
in support forums. USe for 2G Sim Card

  LWiFiClient globalclient ; //Client has to be initiated after WiFi
connection is established.
  client = globalclient;     //This is a temporary solution described in
support forums.
  Serial.println("Connection to Wi-Fi successfull.");
  delay(2000);
}


void loop()
{


  if (millis() >= LastReport + ReportingInterval)   //Send data after about
ReportingInterval (i.e.20 seconds)
  {
  /*
  For more information on Grove sensors:
  http://www.seeedstudio.com/wiki/Grove_System
  */

  delay(2000);  // Sensor readings may also be up to 2 seconds 'old' (its a
very slow sensor)
  //dht.readHT(&t, &h);   //Reading Temperature and humidity from DHT 22

  Serial.println("------------------------------");
  Serial.print("Temperature= "); //Serial monitor print Temperature reading
in Celsius
  Serial.print(hdc.readTemperature());
  Serial.print(" C");

  sprintf(buff,"Battery level= %d", LBattery.level() );
  Serial.print('\n');
```

```
  Serial.println(buff); //Serial monitor print battery level in %

  // Start Ranging -Generating a trigger of 10us burst
  // The sensor is triggered by a HIGH pulse of 10 or more microseconds.
  // Give a short LOW pulse beforehand to ensure a clean HIGH pulse:
  digitalWrite(TRIGPIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIGPIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIGPIN, LOW);

  // Read the signal from the sensor: a HIGH pulse whose
  // duration is the time (in microseconds) from the sending
  // of the ping to the reception of its echo off of an object.
  //Distance calculation
/*****************************************Ultrasonic
sensor*********************************************************
  Distance calculation:
  Velocity of sound in dry air at 20 Â°C (68 Â°F)  = 340 m/s
  1 metre = 100 centimetre            and             1 seconds = 1000000
microseconds
  Velocity of sound = 340 * 100 cm/(1000000 microseconds) = 0.034 cm per us =
(1/29.412) cm per us
  Trash distance (cm) = (Echo pin high time in Âµs/2) x (1/29.1 cm/Î¼s) =
Echo pin high time/58.2 cm
*****************************************************************************
*************************************/
  duration = pulseIn(ECHOPIN, HIGH);
  distance = duration / 58.2;
  /* The speed of sound is 340 m/s or 29 us per cm.The Ultrasonic burst
travels out & back. So to find the distance of object we divide by 58.2  */

  /***** Additional calculation for demo purpose******/
  Level = (distance * 100) / 80;
  //If our trash can was 80cm deep, meaning when the trash reaches 100% it is
at 80cm.
  //This calculation is to output the correct percentage in relation with the
trash depth in cm.

  Serial.print("Trash Level= ");
  Serial.print(Level);
  Serial.println(" %");
  delay(200);

  //Check whether any unauthorised activity is occuring with the device!!
  int activity=0;
  activity=compareResult();
  if(activity==1)
  {
    Serial.println("There is an unusual activity...");
  }
    else Serial.println("No Activity");
    delay(500);

  //String containing all the sensors data according to collection endpoint
API of Ubidots
  //Build the JSON packet according to the format needed by Ubidots
```

```
  //For more info:
http://ubidots.com/docs/api/v1_6/collections/post_values.html

  String payload = "[{\"variable\":\"" VARID1 "\",\"value\":"+
String(Level)+",\"context\":"+Location+"},{\"variable\":\"" VARID2
"\",\"value\":" + String(hdc.readTemperature()) + "},{\"variable\":\"" VARID3
"\",\"value\":" + String(activity) + "},{\"variable\":\"" VARID4
"\",\"value\":" + String(LBattery.level()) + "}]";
  String le = String(payload.length());  // How long is the payload

  //For sending data to Ubidots: http://ubidots.com/docs/api/index.html
  // if you get a connection, report back via serial:

  Serial.print("Connecting to ");
  Serial.println(URL);
  if (client.connect(URL, 80))
  {
    // Build HTTP POST request
    Serial.println("Connection successful with Ubidots server :)");
    client.print(F("POST /api/v1.6/collections/values/?token="));
    client.print(TOKEN);
    client.println(F(" HTTP/1.1"));
    client.println(F("Content-Type: application/json"));
    client.print(F("Content-Length: "));
    client.println(le);
    client.print(F("Host: "));
    client.println(URL);
    client.println();
    client.println(payload);
    client.println();
    client.println((char)26); //This terminates the JSON SEND with a carriage
return
  }
  else
  {
    // if you didn't get a connection to the server:
    Serial.println("Connection Failed");
  }
  delay(100);

  // if there are incoming bytes available
  // from the server, read them and print them:
  if (client.available())
  {
    char c = client.read();
    Serial.print(c);
  }

  // if the server's disconnected, stop the client:
  if (!client.available() && !client.connected())
  {
    Serial.println();
    Serial.println("Disconnecting.");
    client.stop();
  }
  LastReport = millis();
  }
```

```
}


//Function to Store initial axis value of x,y & z for comparision
void axis_initialize()
{
  accel.readXYZ(&x_initial, &y_initial, &z_initial); //read the accelerometer
values and store them in variables  x_initial,y_initial,z_initial
  accel.getAcceleration(xyz_initial);
  ax_initial = xyz_initial[0];
  ay_initial = xyz_initial[1];
  az_initial = xyz_initial[2];
  delay(500);
 }

//To compare initial values with current reading of x,y & z values for
security/unusual protection
int compareResult()
{
  int x,y,z,Xchange,Ychange,Zchange;
  accel.readXYZ(&x, &y, &z); //read the accelerometer values and store them
in variables  x,y,z

  double xyz[3];
  double ax,ay,az;
  accel.getAcceleration(xyz);
  ax = xyz[0];
  ay = xyz[1];
  az = xyz[2];

  if((int)ax!=(int)ax_initial){
      Xchange=1;
  }
  else {
      Xchange=0;
  }

  if((int)ay!=(int)ay_initial){
      Ychange=1;
  }
  else {
      Ychange=0;
  }

  if((int)az!=(int)az_initial){
      Zchange=1;
  }
  else {
      Zchange=0;
  }

  if((Xchange==1)||(Ychange==1)||(Zchange==1)){
    return 1;
  }
  else return 0;
}
```

# Appendix C


# User's Manual

## V1.0

## May 2013

# Index



## 1.0 INTRODUCTION

The HC-SR04 ultrasonic sensor uses sonar to determine distance to an object like bats or dolphins do. It offers excellent non-contact range detection with high accuracy and stable readings in an easy-to-use package. From 2cm to 400 cm or 1" to 13 feet. It operation is not affected by sunlight or black material like Sharp rangefinders are (although acoustically soft materials like cloth can be difficult to detect). It comes complete with ultrasonic transmitter and receiver module.

**Features:**

- Power Supply :+5V DC
- Quiescent Current : <2mA
- Working Currnt: 15mA
- Effectual Angle: <15°
- Ranging Distance : 2cm – 400 cm/1" - 13ft
- Resolution : 0.3 cm
- Measuring Angle: 30 degree
- Trigger Input Pulse width: 10uS
- Dimension: 45mm x 20mm x 15mm

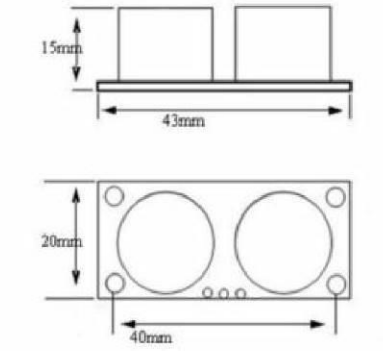
## 2.0 PACKING LIST

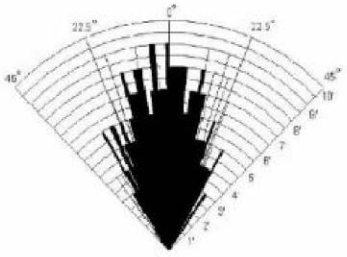

1. 1 x HC-SR04 module

## 3.0 PRODUCT LAYOUT


VCC = +5VDC
Trig = Trigger input of Sensor
Echo = Echo output of Sensor
GND = GND



*Practical test of performance,*
*Best in 30 degree angle*

## 4.0 PRODUCT SPECIFICATION AND LIMITATIONS

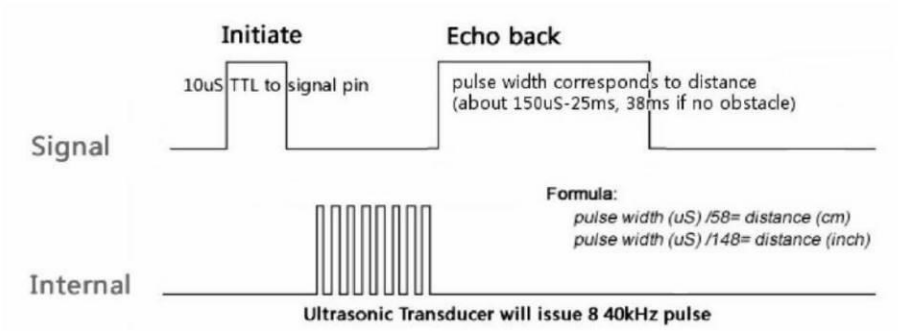
| Parameter | Min | Typ. | Max | Unit |
|---|---|---|---|---|
| Operating Voltage | 4.50 | 5.0 | 5.5 | V |
| Quiescent Current | 1.5 | 2 | 2.5 | mA |
| Working Current | 10 | 15 | 20 | mA |
| Ultrasonic Frequency | - | 40 | - | kHz |

## 5.0 OPERATION

The timing diagram of HC-SR04 is shown. To start measurement, Trig of SR04 must receive a pulse of high (5V) for at least 10us, this will initiate the sensor will transmit out 8 cycle of ultrasonic burst at 40kHz and wait for the reflected ultrasonic burst. When the sensor detected ultrasonic from receiver, it will set the Echo pin to high (5V) and delay for a period (width) which proportion to distance. To obtain the distance, measure the width (Ton) of Echo pin.

Time = Width of Echo pulse, in uS (micro second)

- Distance in centimeters = Time / 58
- Distance in inches = Time / 148
- Or you can utilize the speed of sound, which is 340m/s



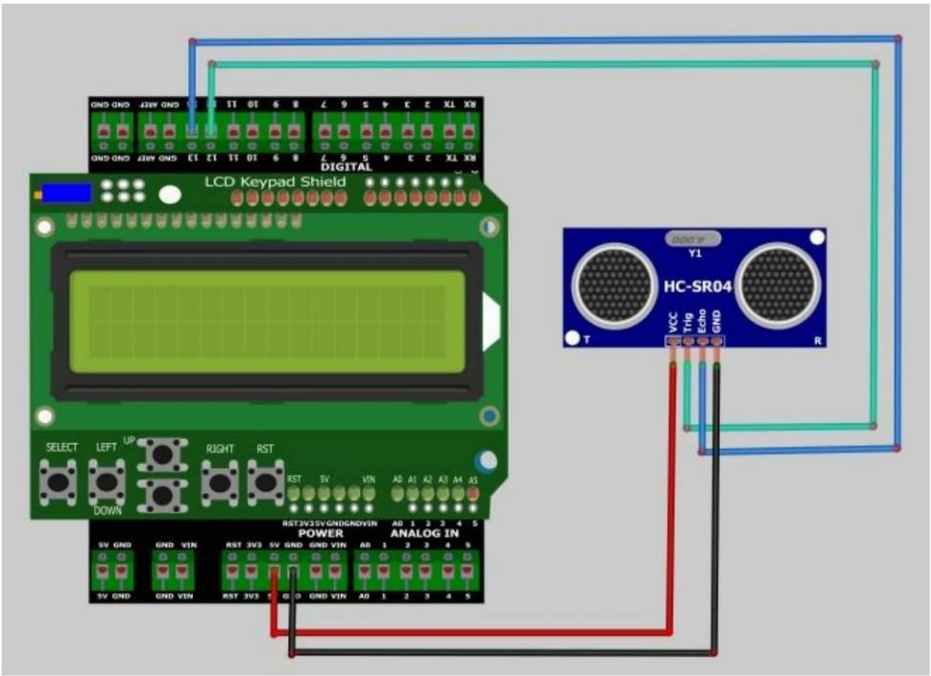

Note:

- Please connect the GND pin first before supplying power to VCC.
- Please make sure the surface of object to be detect should have at least 0.5 meter$^2$ for better performance.

## 6.0 HARDWARE INTERFACE

Here is example connection for Ultrasonic Ranging module to Arduino UNO board. It can be interface with any microcontroller with digital input such as PIC, SK40C, SK28A, SKds40A, Arduino series.



## 7.0 EXAMPLE CODE

This is example code Ultrasonic Ranging module. Please download the complete code at the product page.

```
#include "Ultrasonic.h"
#include <LiquidCrystal.h>
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
Ultrasonic ultrasonic(12,13);

void setup() {
lcd.begin(16, 2);
lcd.setCursor(0, 0);
lcd.print("HC-SR4 testing..");
delay(1000);

}

void loop()
{
  //lcd.clear();
  lcd.setCursor(0, 1);
  lcd.print(ultrasonic.Ranging(CM));
  lcd.print("cm  ");

  delay(100);
}
```

## 8.0 WARRANTY

- Product warranty is valid for 6 months.
- Warranty only applies to manufacturing defect.
- Damaged caused by miss-use is not covered under warranty
- Warranty does not cover freight cost for both ways.

*Prepared by*
***Cytron Technologies Sdn. Bhd.***
19, Jalan Kebudayaan 1A,
Taman Universiti,
81300 Skudai,
Johor, Malaysia.

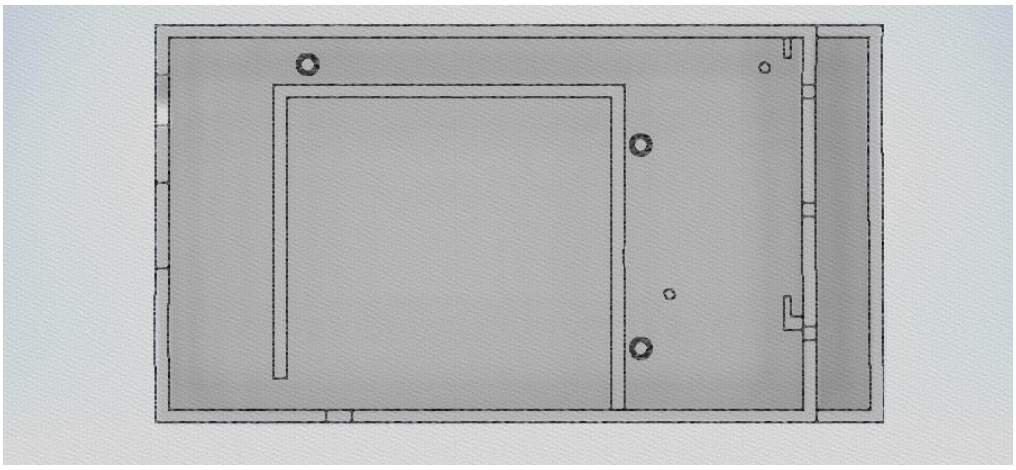
*Tel:* +607-521 3178
*Fax:* +607-521 1861

*URL:* www.cytron.com.my
*Email:* support@cytron.com.my
sales@cytron.com.my

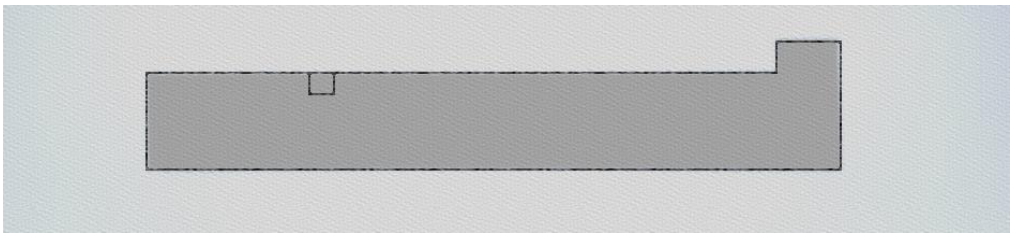
 

# Appendix D

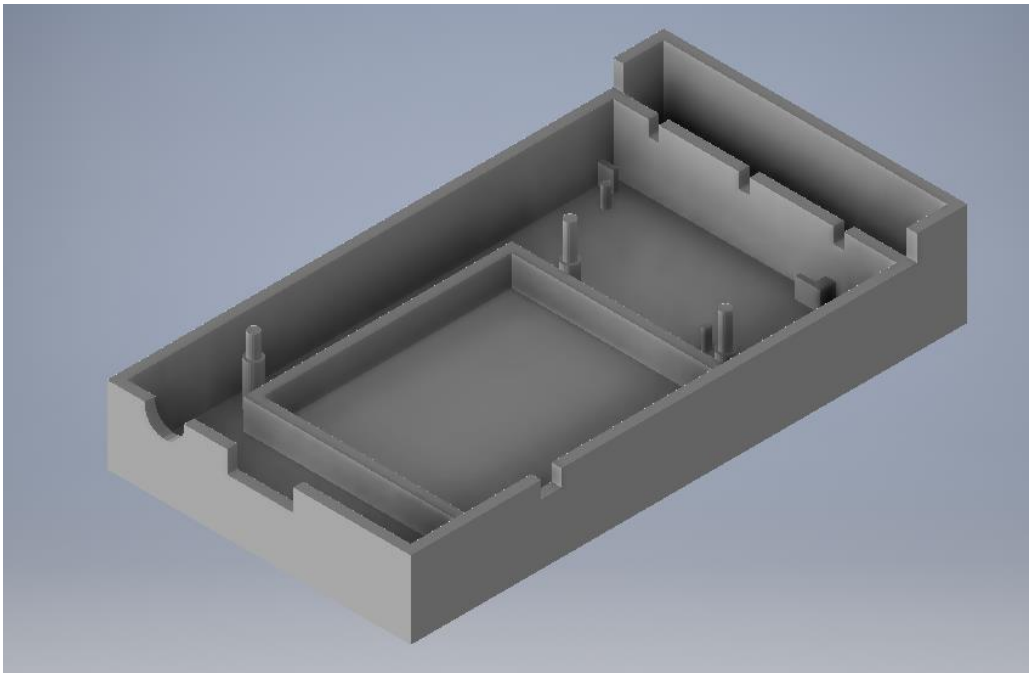
# LinkIt ONE board case



Top View

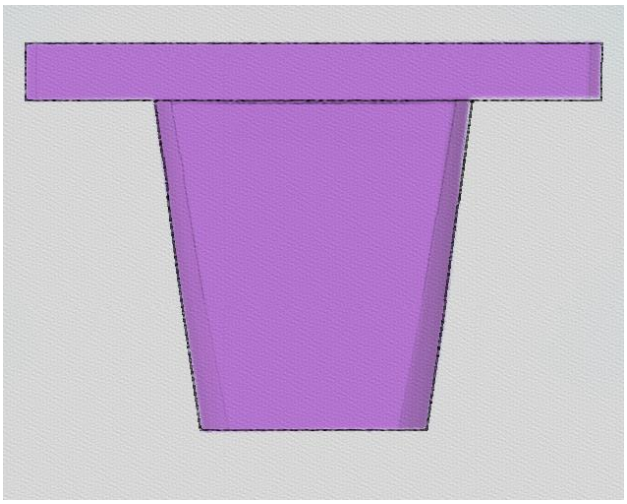


Right Side View

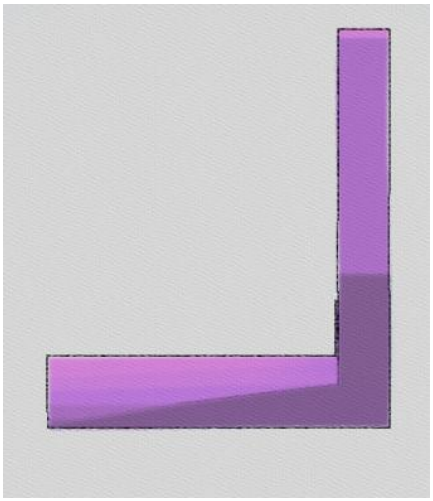


3D render

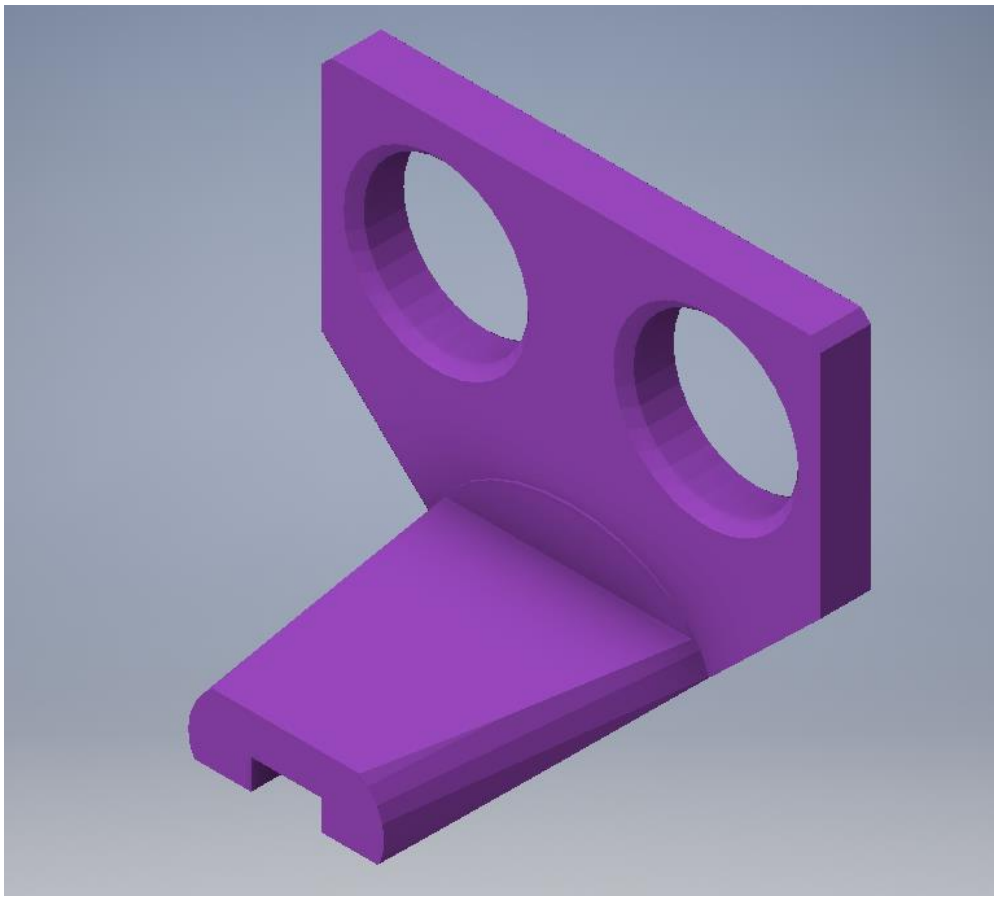
# HC-SR04 Ultrasonic Sensor holder



Top View



Right Side view



3D render