Grass Level Monitor via Image Processing Using Python

Cloud-based IoT Project

Group Members:

Aldrin D. Padua

Paola Andrea Rodriguez-Guzman

Supervisor:

Najeeb Haider Zaidi

Department of Electrical Engineering

1. Summary

The Grass Level Monitor has been designed based on the recommendation of the adviser and the need of the Auckland Council for a system that could remotely monitor grass levels to save human effort from manual evaluation of urban berms and grass fields. The system will help determine if the grasses are tall enough or otherwise, based on the desired parameters.

The common practice for checking and determining if the berms or fields are ready for mowing is that the Auckland Council will send a staff to personally check their sites. Given the fact that they are managing a number of sites, the practice needs some improvement. The grass level monitor will automate the monitoring of the grasses by integrating image processing with the cloud technology and by using the existing and available infrastructures such as the cellular network.

Group was given 20 weeks to design and implement a cloud-based IoT system. Conceptualization of the design and architecture were done on the first three weeks while material management and technical studies were on the next two. The implementation and troubleshooting were done next for 14 weeks. Final documentation was done on the last part.

After 20 weeks, the group was successful in designing and implementing a working cloud-based remote monitoring system called the Grass Level Monitor which works through image processing using Python in both backend and frontend. Amazon Web Services (AWS) served as the cloud platform and the system utilized the cellular network to establish internet connection from the end device to the cloud.

2. Introduction

Background of the project

Grass level monitor is a cloud-based internet of things (IoT) system that was designed to enable remote monitoring of urban berms and grass fields to reduce human effort in grass level evaluation. It makes use of image processing technique to detect grass level, GSM/3G technology to enable internet connection, and cloud technology for the data storage.

The image processing is done through the use of raspberry pi with its camera module as the sensor (together referred to as end device). Together, they will detect the environmental stimulus which is the height of the grass, make decision based on the processed data, and pass that decision onto the cloud. GSM module is used to activate internet connection via GPRS.

Amazon Web Services (AWS) is used as the cloud platform for this project. AWS IoT console provides the mechanism to hook the end device into the cloud while DynamoDB serves as the main database of the system. The AWS API gateway is used as the point of access of the frontend application to the database for data presentation in the frontend. The frontend has been developed through Python language. It pulls data from the API gateway, process them and transform them into a readable and presentable format for the end users to use.

Overall, the project makes use of the already established infrastructures such as the GSM/3G network thru cellular network system and AWS cloud platform to make wireless sensory possible. This, in turn, lessens the cost by cutting the need for newer infrastructures and reduce human effort for the grass level detection, evaluation and monitoring.

Project Requirements

Hardware

- Raspberry Pi Zero W
- Camera module v2
- GSM module: BK-808 v2
- Micro SDHC
- Connecting wires

Software

- Raspbian OS
- AWS IoT console
- AWS DynamoDB
- AWS IAM
- AWS API Gateway

- Laptop
- 5V battery source
- Soldering iron and lead
- Python
- Pi Bakery
- VNC Viewer
- Putty

Human Resource

- Aldrin Padua
- Paola Andrea Rodriguez

Project Design

<u>Image Processing</u>

The image processing algorithm was done through Python. The camera must be positioned strategically and at an angle such as the grass will have a background of lighter colour preferably, the sky. In such condition, the image will be like as shown in Figure 1.

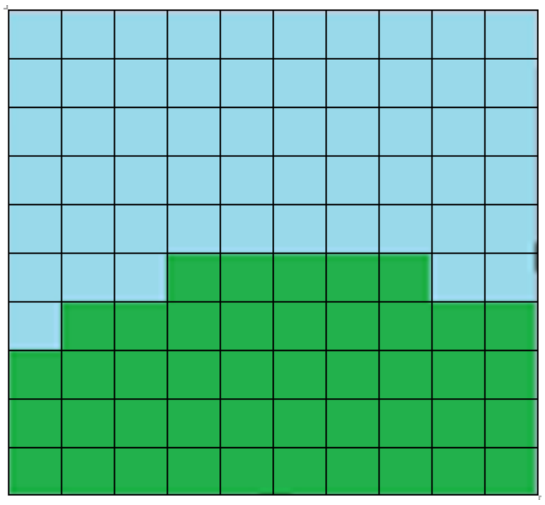
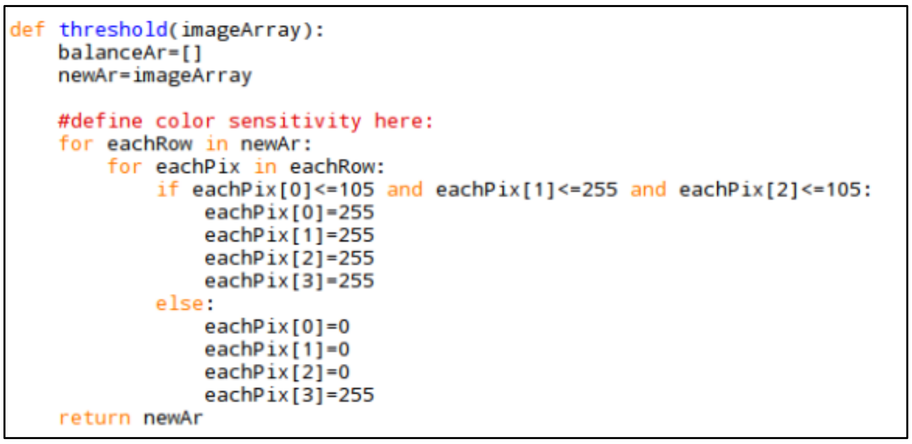


Figure 1. Sample 10x10 Pixel Image

The camera captures 500x500 pixels image by default. But for the sake of simple explanation, this discussion will stick to the example in Figure 1 which is 10x10 pixels. Each and every pixel is a combination of Red, Green and Blue (RGB) colours which, in machine language, is represented by a decimal array ranging from [0 0 0] which is pure black until [255 255 255] which is pure white; the two extremes. That means that any other colour would be represented by array values between pure black and pure white. The colour green on Figure 1 is actually represented as [34 139 34]. Based on trial and error, the default settings of the end device has been set such that if the pixel value is lower than [105 255 105], it will be considered dark. If the pixel is higher, it will be considered light. Below is the part of the code where the color sensitivity and threshold may be adjusted:



```
def threshold(imageArray):
    balanceAr=[]
    newAr=imageArray

    #define color sensitivity here:
    for eachRow in newAr:
        for eachPix in eachRow:
            if eachPix[0]<=105 and eachPix[1]<=255 and eachPix[2]<=105:
                eachPix[0]=255
                eachPix[1]=255
                eachPix[2]=255
                eachPix[3]=255
            else:
                eachPix[0]=0
                eachPix[1]=0
                eachPix[2]=0
                eachPix[3]=255
    return newAr
```

Figure 2. Color Sensitivity and Threshold Definition

(Snippet taken from the image processing code)

To simply the process, the algorithm will convert the image into monochrome so that all pixels will be represented by either just black or white. Conversely, the dark colours will be transformed into white and the light ones will become black. Because the background of the grass is light, it will be black after the image conversion. On the other hand, the grass will be converted to white since its colour is considered dark. Refer to Figure 3.

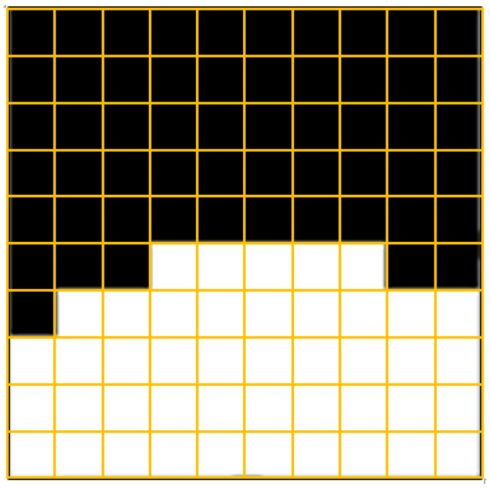
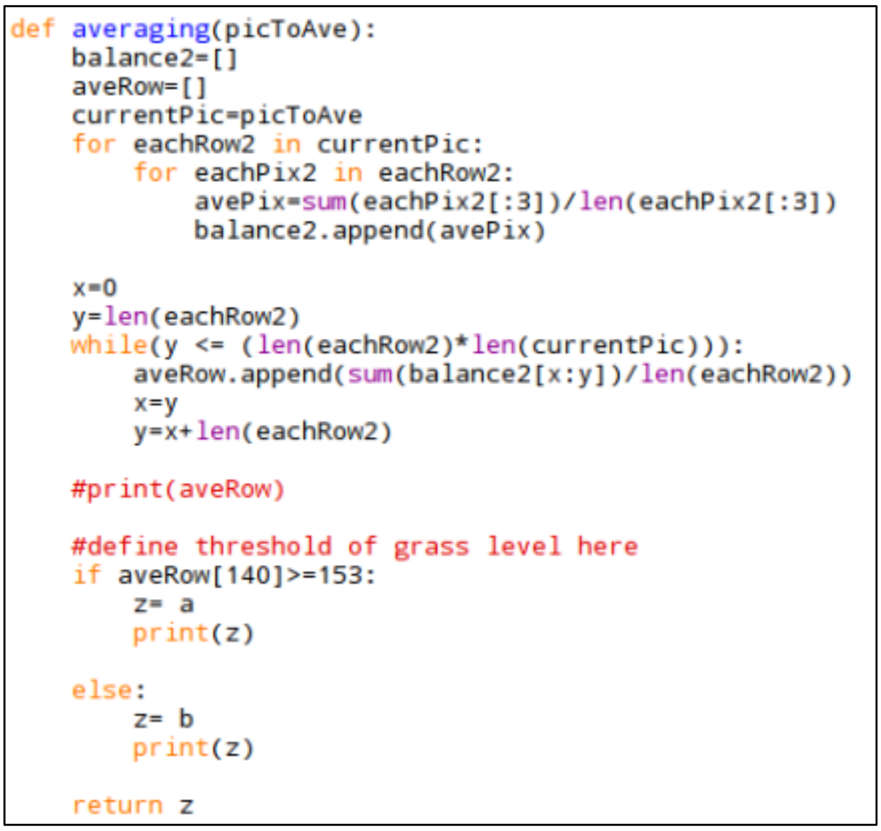


Figure 3. Converted Image

The determining factor by which the device can tell if the grass is tall enough or still short is the intensity of white or black colour in the designated row of pixels in the image. By default, the designated row is row 140. That means, if row 140 has more white colours than black, the grass will be considered tall and ready for mowing. However, if the majority of pixels are black, grass will be considered short. Refer to Figure 4 for the code snippet of the designated row definition.



```
def averaging(picToAve):
    balance2=[]
    aveRow=[]
    currentPic=picToAve
    for eachRow2 in currentPic:
        for eachPix2 in eachRow2:
            avePix=sum(eachPix2[:3])/len(eachPix2[:3])
            balance2.append(avePix)

    x=0
    y=len(eachRow2)
    while(y <= (len(eachRow2)*len(currentPic))):
        aveRow.append(sum(balance2[x:y])/len(eachRow2))
        x=y
        y=x+len(eachRow2)

    #print(aveRow)

    #define threshold of grass level here
    if aveRow[140]>=153:
        z= a
        print(z)

    else:
        z= b
        print(z)

    return z
```

Figure 4. Definition of Designated Row

In the running example, it is assumed that row 7 (R7) is the designated row as shown on Figure 5. Since the majority of pixels in R7 is white, the output of the image process that will be sent to the AWS cloud will be “The grass is now ready for mowing” (Figure 6).

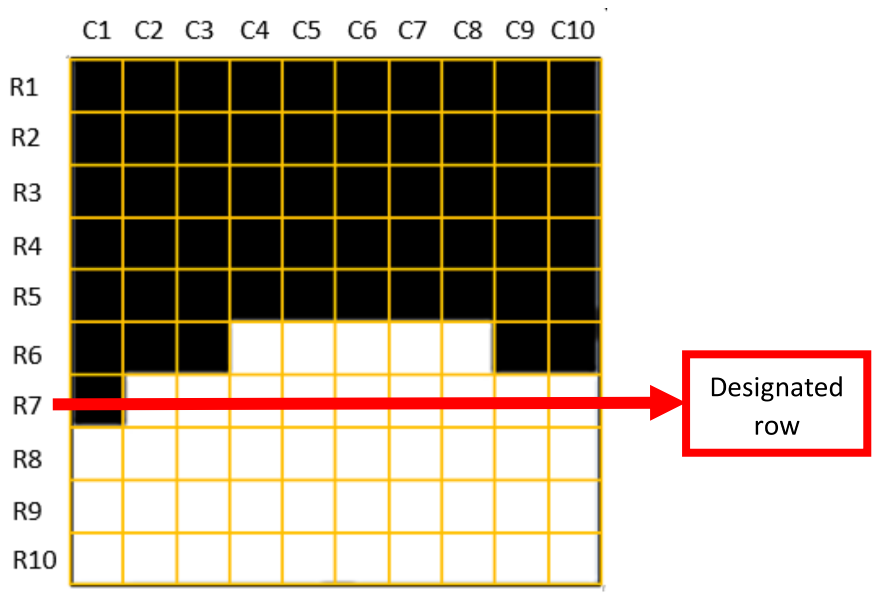
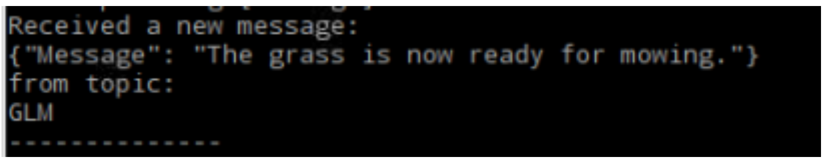


Figure 5. Designated Row

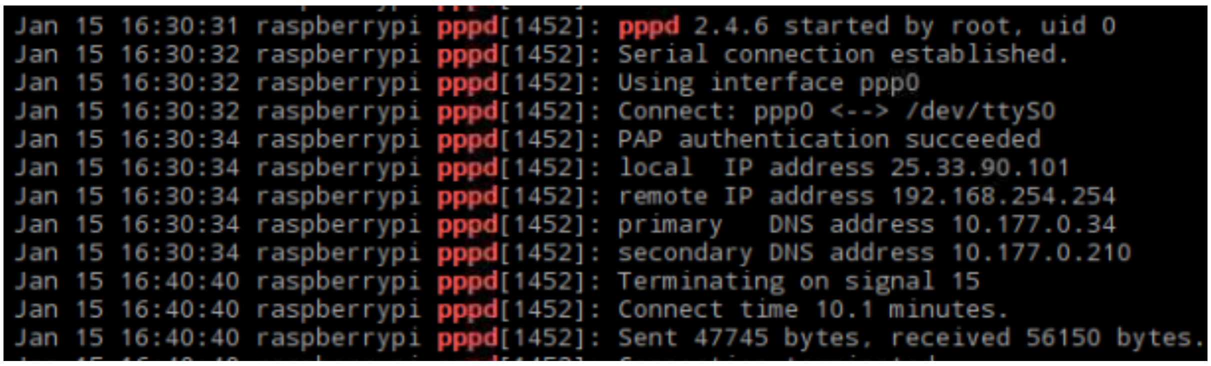


```
Received a new message:
{"Message": "The grass is now ready for mowing."}
from topic:
GLM
--------------
```

Figure 6. Sample Output from the End Device

<u>Setting the GPRS Connection</u>

Setting up a GPRS connection requires a Point-to-Point Protocol (PPP) software. In this project, elinks is used. By changing the Access Point Name (APN) on the rnet file appropriately, the connection can already be started. Below screenshot is a sample establishment of PPP/GPRS connection that has been successfully established using SpeedTalk Network in the United States of America.

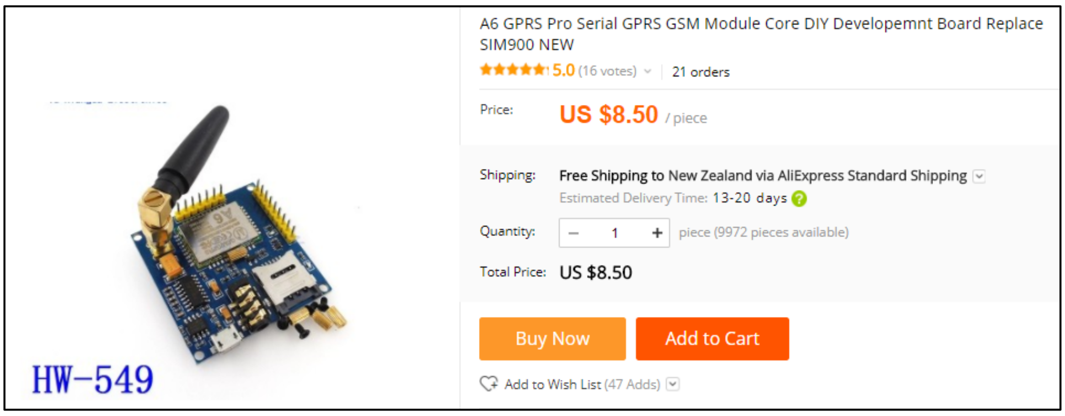


```
Jan 15 16:30:31 raspberrypi pppd[1452]: pppd 2.4.6 started by root, uid 0
Jan 15 16:30:32 raspberrypi pppd[1452]: Serial connection established.
Jan 15 16:30:32 raspberrypi pppd[1452]: Using interface ppp0
Jan 15 16:30:32 raspberrypi pppd[1452]: Connect: ppp0 <--> /dev/ttyS0
Jan 15 16:30:34 raspberrypi pppd[1452]: PAP authentication succeeded
Jan 15 16:30:34 raspberrypi pppd[1452]: local  IP address 25.33.90.101
Jan 15 16:30:34 raspberrypi pppd[1452]: remote IP address 192.168.254.254
Jan 15 16:30:34 raspberrypi pppd[1452]: primary   DNS address 10.177.0.34
Jan 15 16:30:34 raspberrypi pppd[1452]: secondary DNS address 10.177.0.210
Jan 15 16:40:40 raspberrypi pppd[1452]: Terminating on signal 15
Jan 15 16:40:40 raspberrypi pppd[1452]: Connect time 10.1 minutes.
Jan 15 16:40:40 raspberrypi pppd[1452]: Sent 47745 bytes, received 56150 bytes.
```

Figure 7. Established PPP/GPRS connection

(cat /var/log/syslog | grep pppd)

Unfortunately for New Zealand, the GSM module can no longer be used as almost all telecommunication companies have already decommissioned their 2G networks. 2Degrees claims they are yet to decommission theirs but upon trial, the GSM no longer works with their SIM cards. However, this can easily be resolved by replacing the GSM module by 3G although it comes with an expected added cost. Below is a price comparison taken from AliExpress.com for the two modules.



A6 GPRS Pro Serial GPRS GSM Module Core DIY Developemnt Board Replace SIM900 NEW
5.0 (16 votes) | 21 orders

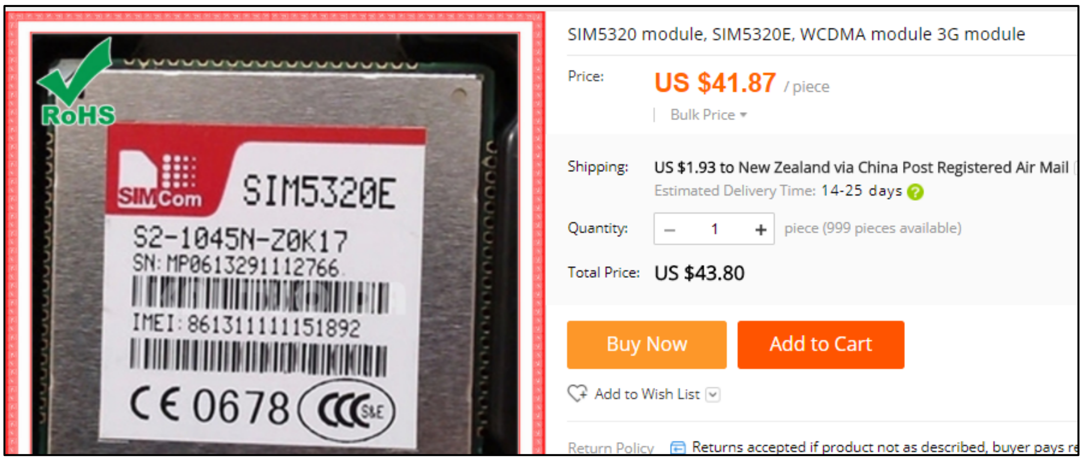
Price: US $8.50 / piece

Shipping: Free Shipping to New Zealand via AliExpress Standard Shipping
Estimated Delivery Time: 13-20 days

Quantity: 1 piece (9972 pieces available)

Total Price: US $8.50

Figure 8. GSM Module (Samiore Robot, n.d.)



SIM5320 module, SIM5320E, WCDMA module 3G module

Price: US $41.87 / piece

Bulk Price

Shipping: US $1.93 to New Zealand via China Post Registered Air Mail

Estimated Delivery Time: 14-25 days

Quantity: 1 piece (999 pieces available)

Total Price: US $43.80

Buy Now Add to Cart

Add to Wish List

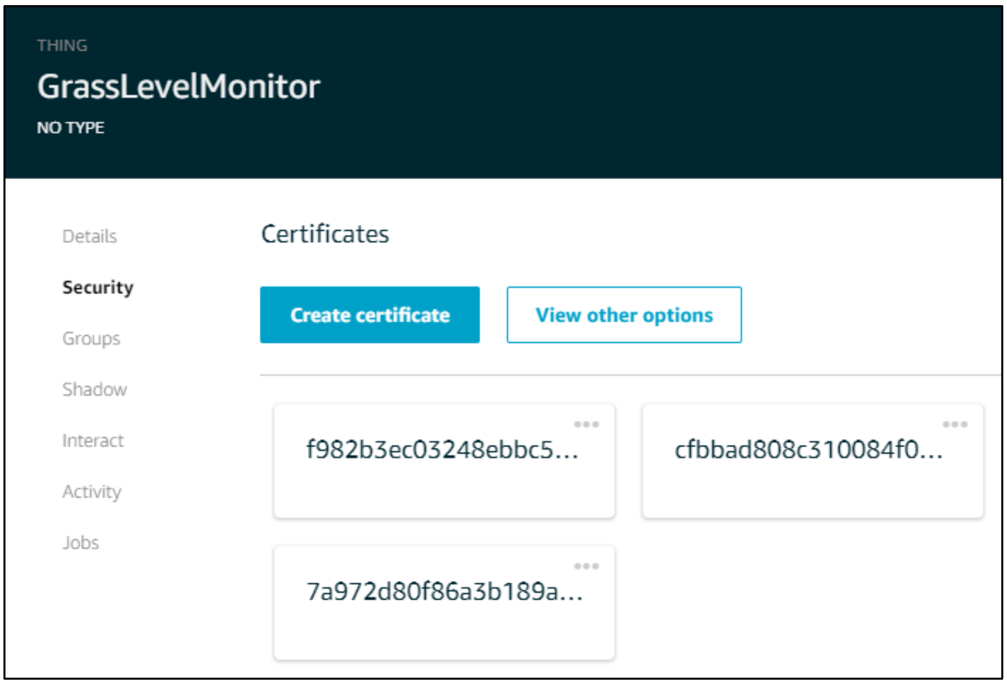
Return Policy Returns accepted if product not as described, buyer pays r

Figure 9. 3G Module (Kai Tuo Da, n.d.)

For the sake of demonstration purposes and as per adviser, the group will just use the WiFi connection in replacement of the GPRS due to lack of budget and time constraint.

AWS Cloud Integration

On AWS IoT Console, the certificates and private keys have been generated. These certificates and private keys were installed on the Raspberry Pi to connect the device to the Cloud.



THING

GrassLevelMonitor

NO TYPE

Details

Security

Groups

Shadow

Interact

Activity

Jobs

Certificates

Create certificate View other options

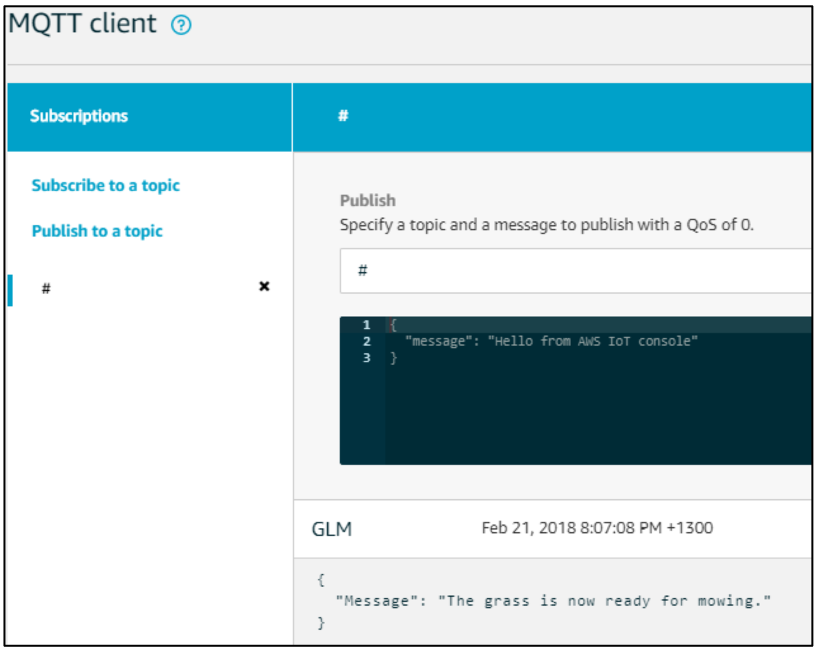
f982b3ec03248ebbc5…

cfbbad808c310084f0…

7a972d80f86a3b189a…

Figure 10. AWS IoT Certificates

Installation of the keys into the end device enables the Raspberry Pi to talk with the MSQTT client of the IoT console. That means, the output of the image processing is now conveyable to the cloud. Refer to Figure 11 for a sample message that has been transmitted into the cloud via MSQTT.



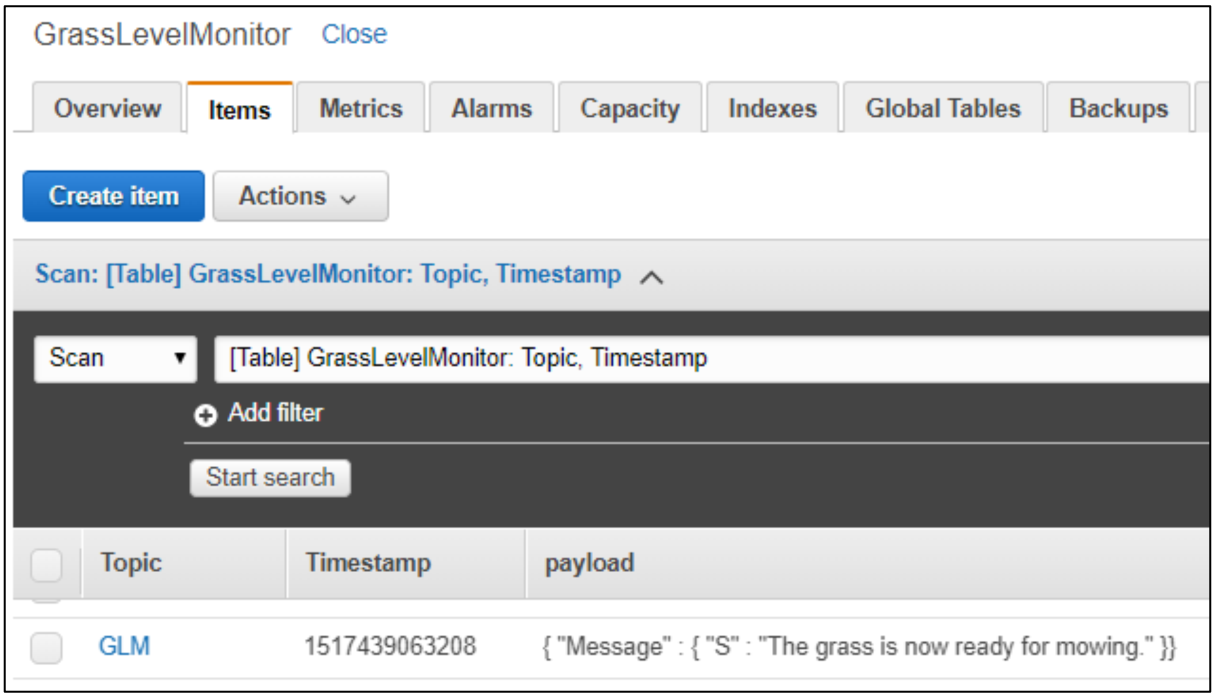
```
{
  "message": "Hello from AWS IoT console"
}
```

GLM Feb 21, 2018 8:07:08 PM +1300

```
{
  "Message": "The grass is now ready for mowing."
}
```

Figure 11. Sample Message Received by the MSQTT Client

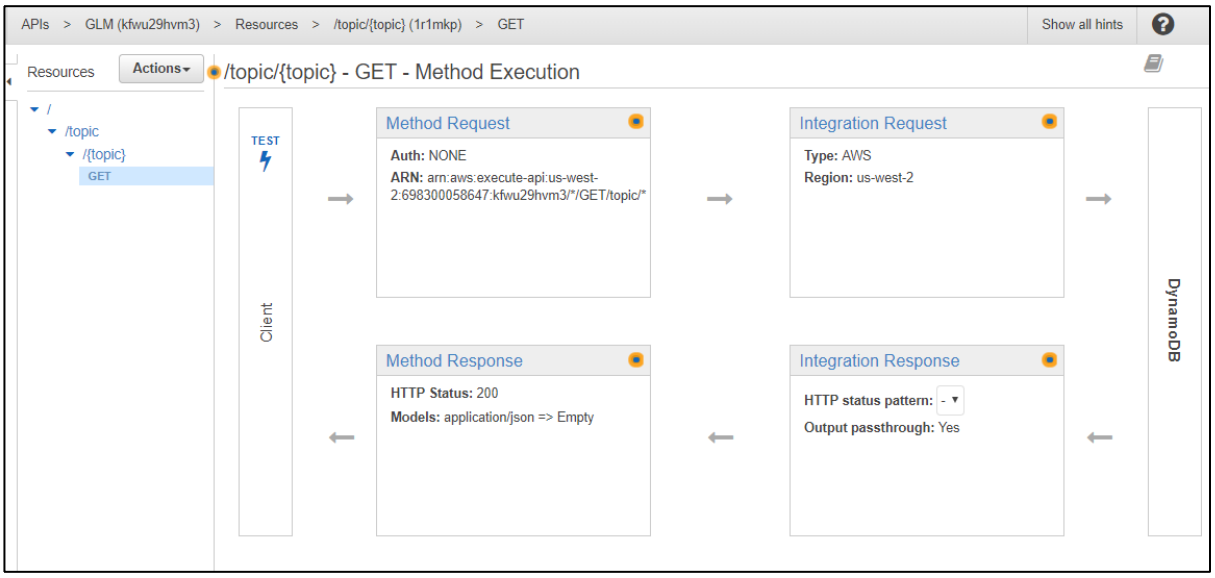
Once data reaches the MSQTT client, the message will be directed and placed inside the table in DynamoDB. The IoT console has been configured with a rule to pass every message that is received directly to the database in JSON format.



Topic	Timestamp	payload
GLM	1517439063208	{ "Message" : { "S" : "The grass is now ready for mowing." }}

Figure 12. Sample Database Entry

On the same hand, AWS API gateway has been configured such that it pulls out every entry on the DynamoDB table and give them their own unique URL using the table and topic name as basis.



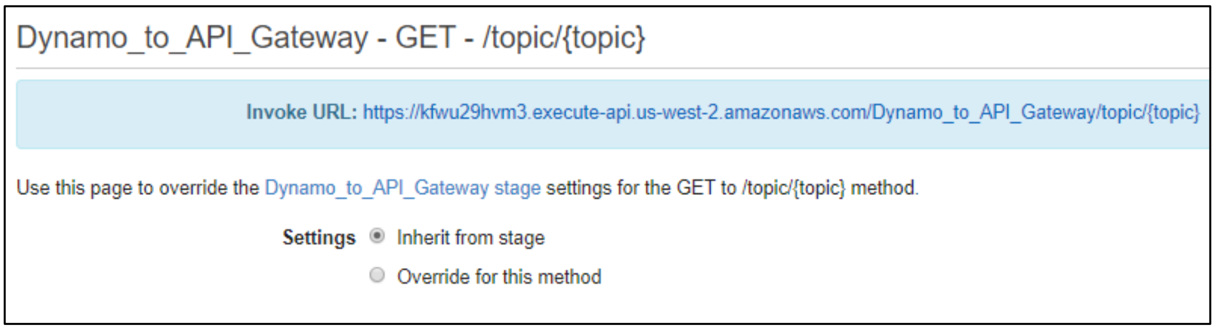
Method Request
Auth: NONE
ARN: arn:aws:execute-api:us-west-2:698300058647:kfwu29hvm3/*/GET/topic/*

Integration Request
Type: AWS
Region: us-west-2

Method Response
HTTP Status: 200
Models: application/json => Empty

Integration Response
HTTP status pattern: -
Output passthrough: Yes

Figure 13. API Gateway



Dynamo_to_API_Gateway - GET - /topic/{topic}

Invoke URL: https://kfwu29hvm3.execute-api.us-west-2.amazonaws.com/Dynamo_to_API_Gateway/topic/{topic}

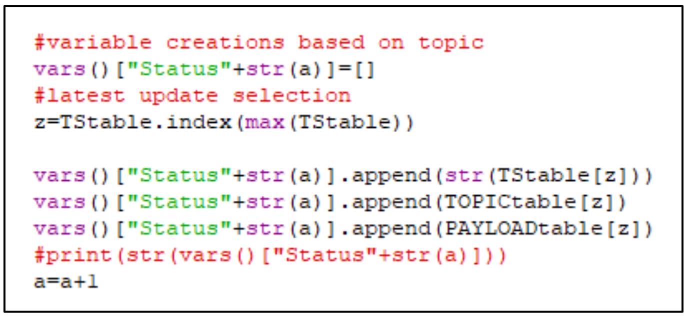
Use this page to override the Dynamo_to_API_Gateway stage settings for the GET to /topic/{topic} method.

Settings [✓ Inherit from stage] Override for this method

Figure 14. API Gateway URL

Data Presentation

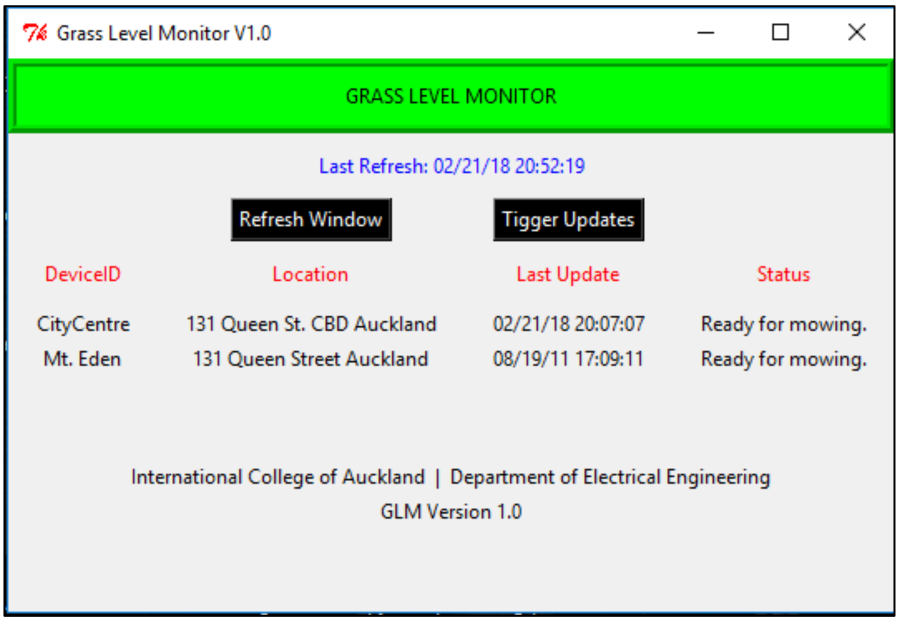
The frontend application has been developed using Python Tkinter. The application pull the data from the API gateway then transforms the JSON strings into a more presentable format. Among all the entries available in the API gateway, the application is programmed to select only the latest update.



```
#variable creations based on topic
vars()["Status"+str(a)]=[]
#latest update selection
z=TStable.index(max(TStable))

vars()["Status"+str(a)].append(str(TStable[z]))
vars()["Status"+str(a)].append(TOPICtable[z])
vars()["Status"+str(a)].append(PAYLOADtable[z])
#print(str(vars()["Status"+str(a)]))
a=a+1
```

Figure 15. Latest Update Selection Algorithm


Grass Level Monitor V1.0

GRASS LEVEL MONITOR

Last Refresh: 02/21/18 20:52:19

Refresh Window | Tigger Updates

DeviceID	Location	Last Update	Status
CityCentre	131 Queen St. CBD Auckland	02/21/18 20:07:07	Ready for mowing.
Mt. Eden	131 Queen Street Auckland	08/19/11 17:09:11	Ready for mowing.

International College of Auckland | Department of Electrical Engineering

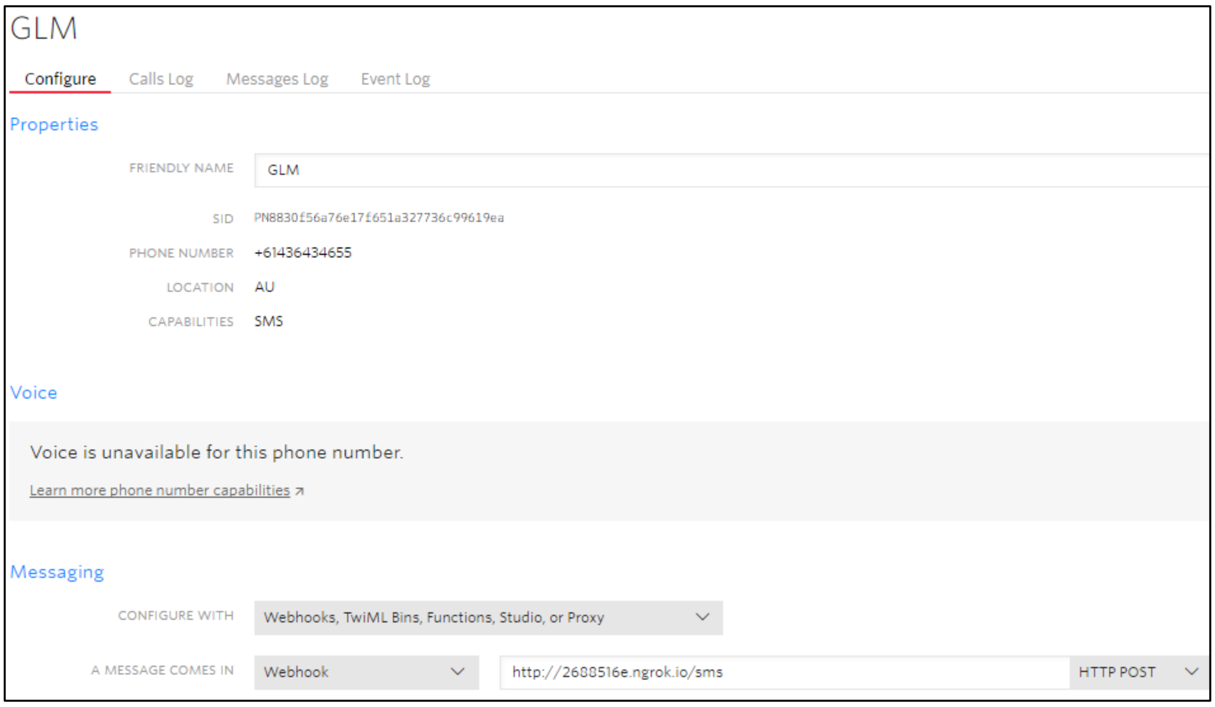
GLM Version 1.0

Figure 16. Frontend Application: Grass Level Monitor

<u>SMS Trigger</u>

Going back to the end device, the image process algorithm have two ways of being triggered. Frist, there is a scheduled weekly run by default. Second, it can be triggered by an SMS sent to the device's phone number. Twilio is an application installed on the Raspberry Pi which allows it to have a phone number associated with itself. Ngrok, on the other hand, is another application which creates a tunnel from the end device to a local server. It also generates a URL which is unique for the device. So basically, it functions like the API Gateway of AWS. The URL can be associated with a domain name so it won't have to change every time Ngrok is restarted. However, it has a subscription fee which should be paid in full annually. So for the purpose of demonstration, this project will just make use of the dynamic URL.

The URL generated by Ngrok will be configured on the Twilio website where the phone number of your device can be managed. Below is a glimpse of the configuration of the Grass Level Monitor phone number on Twilio website via webhook.


GLM

Configure Calls Log Messages Log Event Log

Properties

FRIENDLY NAME GLM

SID PN8830f56a76e17f651a327736c99619ea

PHONE NUMBER +61436434655

LOCATION AU

CAPABILITIES SMS

Voice

Voice is unavailable for this phone number.

Learn more phone number capabilities

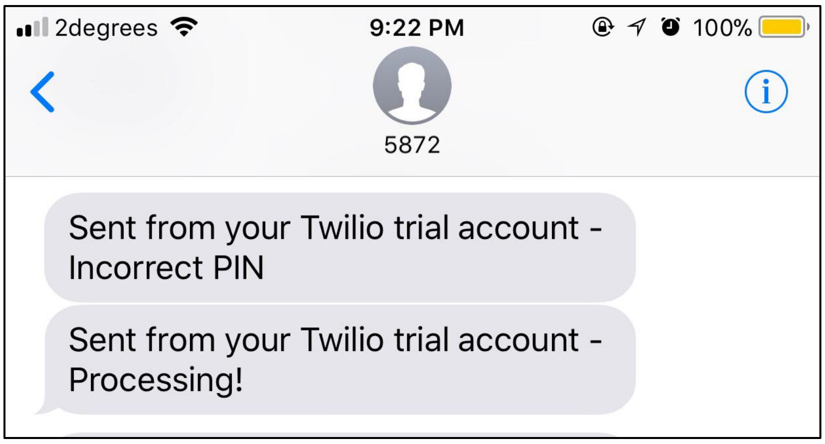
Messaging

CONFIGURE WITH Webhooks, TwiML Bins, Functions, Studio, or Proxy

A MESSAGE COMES IN Webhook http://2688516e.ngrok.io/sms HTTP POST

Figure 17. URL Configuration on Twilio Phone Number

While running, the Raspberry Pi always listens to HTTP Port 5000 where the tunnel created by Ngrok is running and where Twilio sends its messages. Upon reception of the SMS, the device is programmed to reply “Processing!” and start the image processing if the PIN is correct. On the other hand, it is programmed to reply “Incorrect PIN” and do nothing if the PIN is not right.


2degrees 9:22 PM 100%

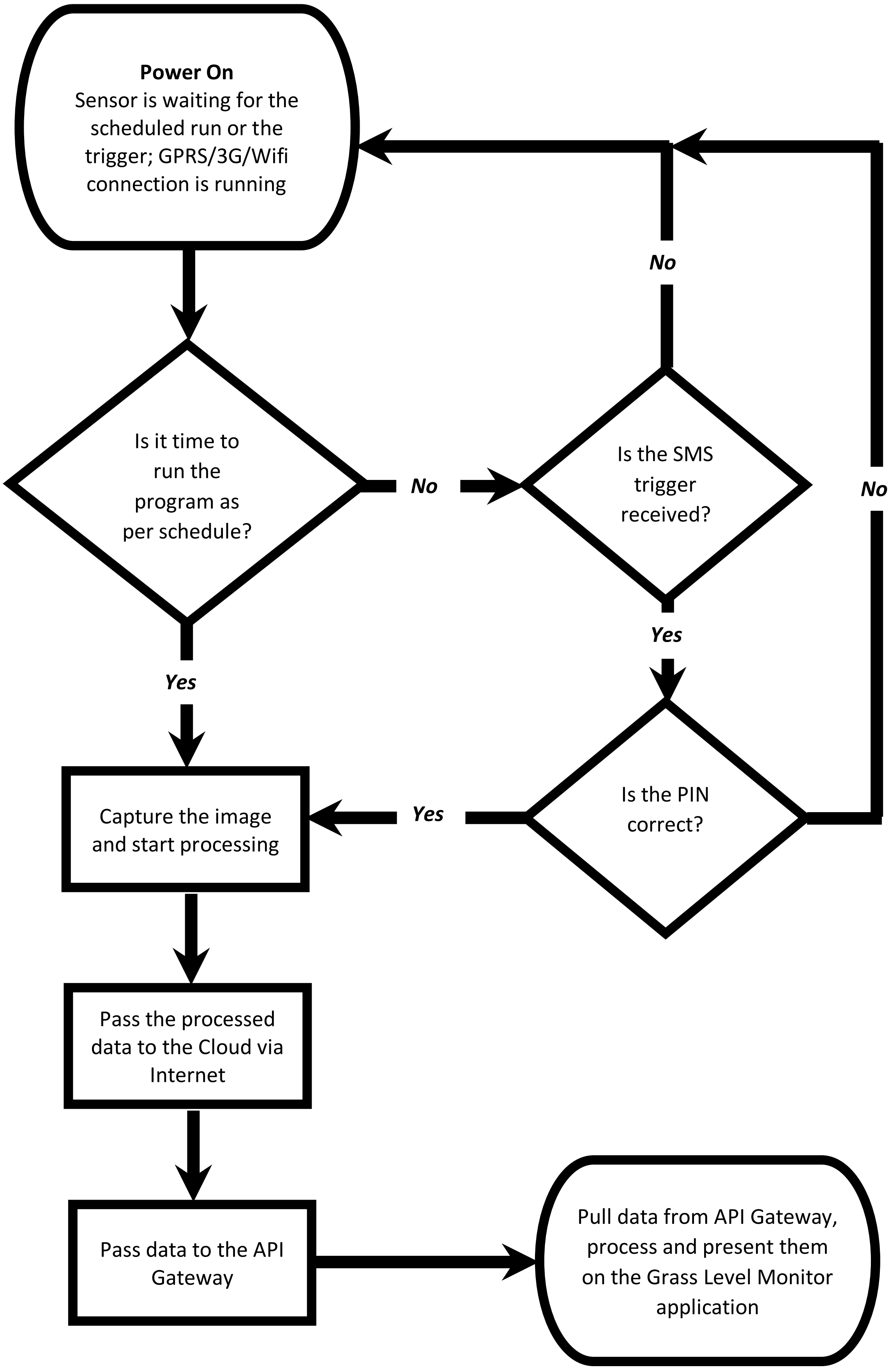
5872

Sent from your Twilio trial account - Incorrect PIN

Sent from your Twilio trial account - Processing!

Figure 18. Sample Replies from the End Device

Flow Chart


Power On
Sensor is waiting for the scheduled run or the trigger; GPRS/3G/Wifi connection is running
Is it time to run the program as per schedule?
No
Is the SMS trigger received?
No
No
Yes
Yes
Is the PIN correct?
Yes
Capture the image and start processing
Pass the processed data to the Cloud via Internet
Pass data to the API Gateway
Pull data from API Gateway, process and present them on the Grass Level Monitor application

3. Discussion

The group have had a great exposure to Python programming form backend to frontend. Both members have previous experience with computer programming languages such as HTML and C++. Although Python is another language, the structure of the commands and syntax are pretty much similar. Programs were optimized and made flexible so changes made on the system will not require entire program rewrite. They were also made as structural as the group is capable of.

Cloud integration with AWS platform was a bit of a challenge since the group is new to that technology. However, with constant familiarization, research and study, members were able to explore and learn about a number of functions and modules that AWS can offer. The team had a successful experience especially in the configuration of AWS IoT console, DynamoDB, IAM, Simple Notification Service (SNS) and API Gateway. SNS can be used as alternative for SMS but Twilio does not currently support 4-digit coded numbers such as the one that AWS SNS uses. However, the group already prepared the system for SNS service should Twilio decide to support short coded numbers in the future.

GSM module was easy to install, use and integrate with the Raspberry Pi. GRPS connection was successfully established in the USA using SpeedTalk Network brand. The real challenge was upon arrival to New Zealand, the GSM does not work. Spark and Vodafone said they no longer support GSM and they have already decommissioned their 2G network. 2Degress said they are yet to ditch their 2G but upon trial, the GSM module was still not functional with their network. The solution proposed was to upgrade to 3G. However, due to time constraint and budget limitations, the group has been advised to use WiFi instead for demo purposes.

After the successful completion of the project, the group is planning to take AWS Cloud trainings and certifications to explore new horizons and improve their current skillset. AWS cloud technology and IoT are booming fields of technology that the members are ready to venture on.

4. Evaluation

4.1 Self-Evaluation (Aldrin Padua)

My programming skills have been improved and I had a very great exposure to Python. I have always been fascinated with the Cloud technology especially that of AWS and now I have experienced it first-hand and successfully implemented our own project with it. My microcontroller programming was also enhanced as I was trained very well with the Raspberry Pi. The project was a difficult for IoT amateurs like us but with team work we were able to complete it. I was pushed to work on timelines and motivate myself harder to meet our course requirements.

4.2 Peer Evaluation (Paola Rodriguez)

Paola was very helpful especially in troubleshooting our programs. She was very keen to details and was a great help. She also did great on the AWS cloud integration and configuration which was her part of the workload.

5. Conclusion

The Grass Level Monitor has been successfully implemented. The problem with the 2G network has been worked out via WiFi. In real-life deployment, 3G module will be used instead. The AWS Cloud is functioning properly and no issues has been found. The system uses existing infrastructures such as cellular network and AWS cloud platform to carry out its function. This project is highly versatile and can easily adapt to perform different functions. In the future it can be modified and used for remote water-level monitoring, air pollution monitoring, and a lot more other.

6. References

Samiore Robot. (n.d.). A6 GPRS Pro Serial GPRS GSM Module Core DIY Development Board Replace SIM900 NEW [Photograph]. Retrieved from https://www.aliexpress.com/item/A6-GPRS-Pro-Serial-GPRS-GSM-Module-Core-DIY-Developemnt-Board-Replace-SIM900-NEW/32816930262.html?ws_ab_test=searchweb0_0,searchweb201602_5_10152_10151_10065_10344_10068_10342_10547_10343_10340_10548_10341_10084_10083_10618_10630_10307_5722317_5711211_10313_10059_10534_100031_10629_10103_10626_10625_10624_10623_10622_10621_10620_10142,searchweb201603_1,ppcSwitch_5_ppcChannel&algo_expid=121cdafb-fb69-4c0b-a9d6-dcc0c1185d62-6&algo_pvid=121cdafb-fb69-4c0b-a9d6-dcc0c1185d62&priceBeautifyAB=0

Kai Tuo Da. (n.d.). *SIM5320 module, SIM5320E, WCDMA module 3G module* [Photograph]. Retrieved from https://www.aliexpress.com/item/SIM5320-module-SIM5320E-WCDMA-module-3G-module/32806544021.html?ws_ab_test=searchweb0_0,searchweb201602_5_10152_10151_10065_10344_10068_10342_10547_10343_10340_10548_10341_10084_10083_10618_10630_10307_5722317_5711211_10313_10059_10534_100031_10629_10103_10626_10625_10624_10623_10622_10621_10620_10142,searchweb201603_1,ppcSwitch_5&algo_expid=d348457a-60fa-4c3b-a7f4-6ed4306e12ed-5&algo_pvid=d348457a-60fa-4c3b-a7f4-6ed4306e12ed&priceBeautifyAB=0