Smart Solar Oven

A Senior Project

presented to the

Faculty of the Computer Engineering Department

California Polytechnic State University, San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Bachelor of Science

by

Aaron Gragg

Advised by Bryan Mealy

March, 2015

© 2015 Aaron Gragg

Table of Contents

Abstract

Though solar oven technology provides a green and cost effective means of cooking, it is often not adopted because it fails to meet the demands of speed and reliability that are commonplace in our society. This project explores the modification of a traditional box solar to meet these demands by allowing the oven to report temperature to a user's phone as well as orient itself towards the sun throughout the day.

Introduction

Solar ovens provide a sustainable and cost effective means of cooking. Their presence throughout the world has increased as humanitarian groups, governments, and individuals shift focus towards the growing energy concerns. In India, for example, 311 million citizens live without access to electricity[1]. In the last several years, roughly $600 million has been spent by the Government of Indian to equip the next generation with the knowledge and tools to harness solar energy for preparing food[2]. With the wellbeing of much of the world fated on our ability to harness sustainable energies, the design of solar ovens and the culture behind their use is of great importance.

For those who have access to conventional gas or electric ovens, using a solar alternative is slower and less reliable. Reaching a good cooking temperature may take a few hours and regular monitoring to adjust the facing of the oven. Also, if you have left your oven unattended, you cannot rely on your food being ready when you come back. In order to incentivize this green alternative, this project attempts to alleviate some of these issues.

In the course of this project, I discovered that many of my peers had not been introduced to the concept of a solar oven. To clarify, a solar oven, or sun oven, is a device that reflects and traps light in order to heat a cooking chamber. There are currently many different existing designs. One of the most popular designs is a simple box oven. This design uses fans or wings to reflect light into the center box shaped cooking area. Another popular design is a parabolic dish that reflects light into a very fine focal point. For this project, I chose to use a variation of a box oven.

[1] http://documents.worldbank.org/curated/en/2014/10/20346115/power-all-electricity-access-challenge-india

[2] http://solarcooking.wikia.com/wiki/India

Project Objectives

The objectives of this project are threefold. First, the oven should generate all necessary electrical power from the sun. This is necessary as to not compromise the benefits of using a solar oven.

Second, be able to adapt a solar oven to report temperature to a user's device. The user should be able to specify an interval rate of temperature reports remotely. The oven should also rest on a rotating platform that detects and aligns itself to an optimal cooking angle. The temperature reports will improve reliability by providing a guarantee when an acceptable temperature has been reached. The rotating base will improve functionality by no longer requiring manual reorienting of the oven. Together, these alterations should adapt the solar oven to be a more viable option for those of us working away from home in the day.

Third, with this new solar oven, data can be collected to compare the effectiveness of varying reorientation intervals. This could lead to some conclusive evidence to use in a cost-benefit analysis of the oven as a product.

System Design

The final product, pictured in Figure 1, consists of the following subcomponents: oven, microcontroller, temperature measurement, light detection, motor, and power. Descriptions of each subsystem, as well as a cost breakdown, are located below. For information on subcomponent integration, refer to the System Diagram (Figure 2).

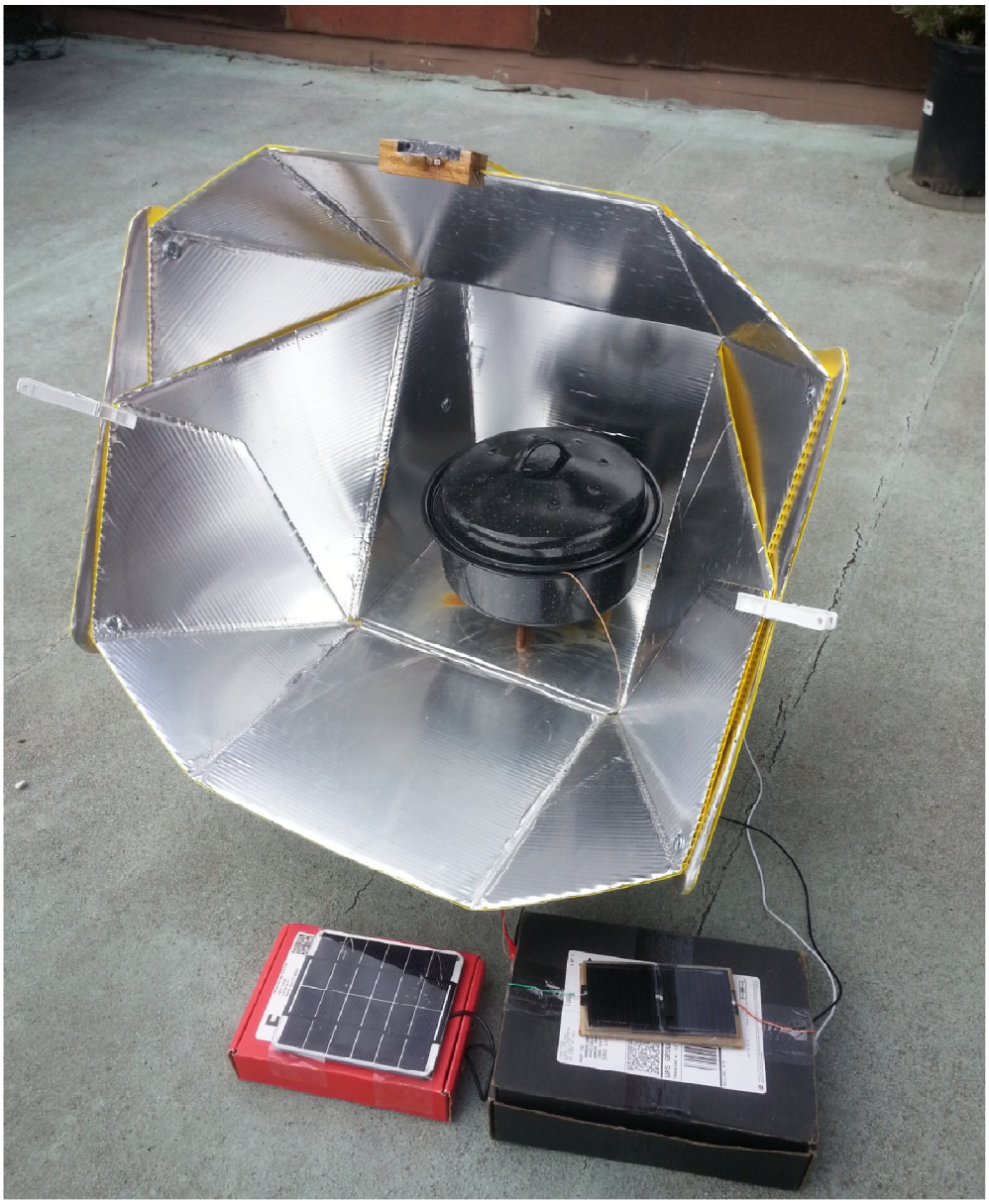

Figure 1: Final Product

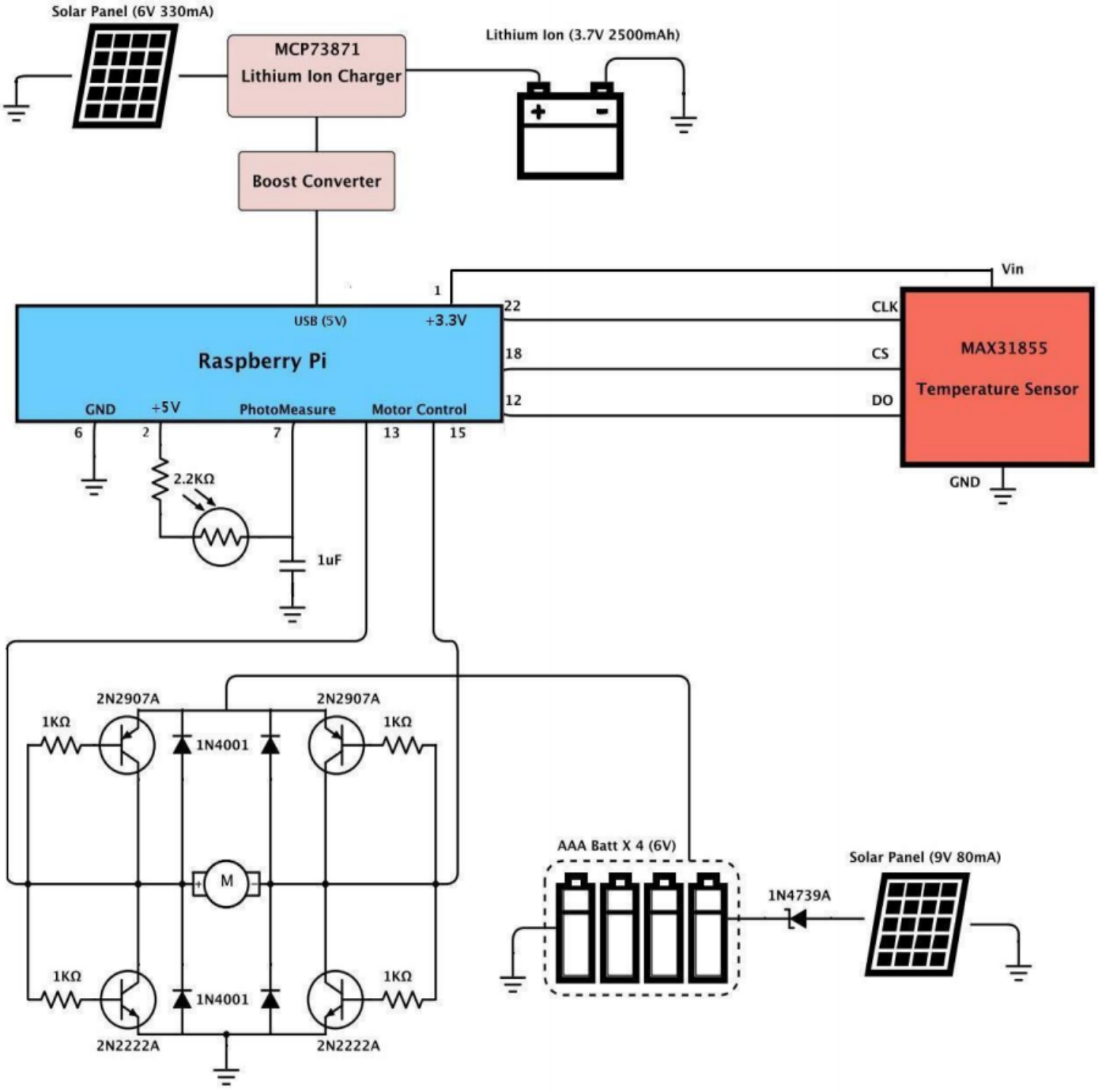

Figure 2: System Diagram

The Oven

The oven chosen for this project was the All Season Solar Cooker from Solar Cooker Biz, as seen in Figure 1. This oven was chosen for several reasons. It is lightweight (~8 lbs) and therefore would not require as strong a motor to turn. It has a cooking area that is large enough for any standard cooking dish. It also comes with a small device, which is used to determine the optimal cooking angle. The device is a small wooden peg that attaches to the hood of the oven. The direction of the shadow being cast by the peg indicates to the user which angle is best for cooking. This device was adapted to help autonomously determine the optimal cooking angle and is discussed later.

Microcontroller and Communication

A Raspberry Pi with an Edimax WiFi dongle was used to create a means of communication with the user. Once the Pi was online, it accepts commands by checking a Gmail account (graggPi@gmail.com). In this way, a user could send emails to control the oven. Implemented commands are listed at the bottom of this section. Additionally, a simple email to text service was used to allow the Pi to send texts of temperature data back to the user through the same Gmail account. An example use of operation is as follows: The user turns the Pi on, sends emails to specify the rate of temperature reports sent via text (REPORT) and rotation of oven (ANGLE), receives text messages on temperature data regularly throughout the day, and sends email to cancel the oven operations (CLEAR) once cooking is completed. For Python code, see checkGmail.py in the Appendix.

- Commands (Subject Line):
    - REPORT - send temperature data immediately
    - REPORT:X - send temperature data at regular X minute intervals

- ANGLE – rotate to find optimal cooking facing
- ANGLE:X – rotate to find optimal cooking facing at regular X minute intervals
- CLEAR – end all repeating commands

Temperature Measurement

Traditional IC temperature sensors are only reliable below 200° F. For this reason, the MAX31855 Thermocouple Sensor with Adafruit's Thermocouple amplifier breakout board was used to determine cooking temperatures. This module is effective between -200°C to 1350°C. Adafruit distributes a library to easily control the CLK, CS, and DO lines of the sensor. For implementation of this library and code to send the temperature data as a text over email, see textTemp.py in the Appendix.

Light Detection

In order to detect the optimal cooking angle, four photoresistors were affixed to the supplied device. The result is shown in Figure 3 below. Taking a reading across these resistors proved slightly difficult as the Raspberry Pi only comes with digital input. To work around this constraint, the following technique is used. The PhotoMeasure pin is set output low then immediately set as an input. This first grounds the capacitor, emptying it of charge, and then allows it to charge back up. For relative information on the photoresistors rating, we watch the time it takes to toggle the PhotoMeasure input to a digital high. Code for this process is in the RCtime() function in findAngleGM.py in the Appendix.

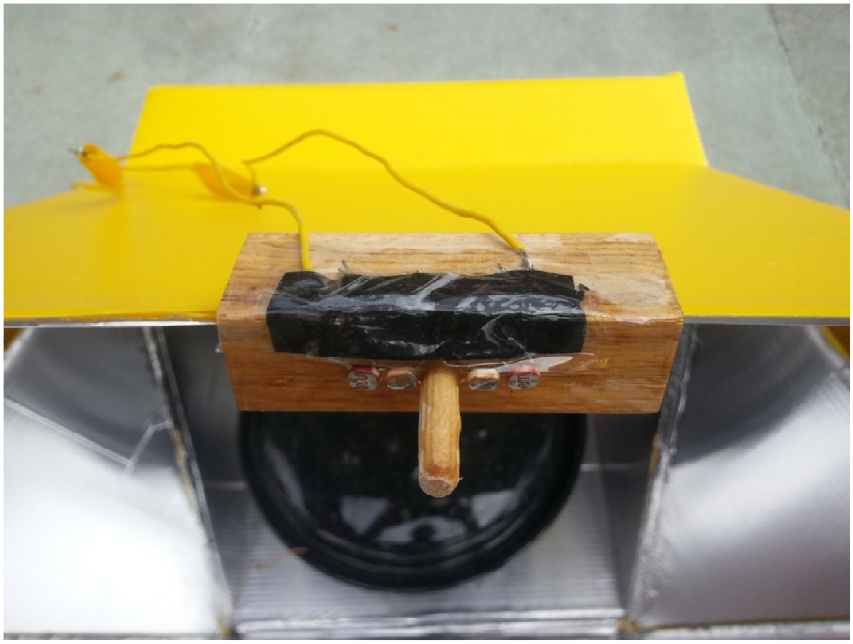

Figure 3: Light detection device

Motor Control

For this kind of high torque application a gearmotor was much more suited than a servo. The gearmotor used in this project has a gear ratio of 1500:1 and a stall torque of 1497 oz-in at 12V. For this kind of continuous rotation motor, changing direction requires flipping the voltage on the leads. In order to provide this control from the Pi, an H-bridge with NPN and PNP transistors was used. Now, with motor control and a gauge of relative cooking intensity, the oven can be rotated to find the best orientation for cooking. For code of this process, see findAngleGM.py in the Appendix. Figure 4 below shows the motor and base assembly that was used.

Figure 4: Motor and base assembly

Solar Power

In order to power this project from the sun, two separate rechargeable battery and solar cell systems were used. The power for the Pi and thermocouple was derived from a 6V 2W solar panel and stored in a lithium ion battery with a capacity of 2500mAh. In order to protect the battery, the MCP73871 controlled battery charging. Also, to ensure 5V at 500mA to the Raspberry Pi, Adafruit's USB PowerBoost 500 Basic was used. The power for the motor was generated from two 4.5V 0.36W solar cells in series powering four rechargeable AAA's. Though there is no smart charging IC used here, the rate of charge is roughly C/10 and is therefore considered safe from accidental overcharge. A diode is used to ensure the solar cells do not suffer reverse current.

Initially, experiments were conducted to attempt to avoid battery usage. It was found, however, that the current pull from the Pi was spermatic depending on the use of certain WiFi operations and that the motor simply required more power than could be instantaneously supplied.

Cost

Product	**Units**	**Approximate Unit Cost**	**Cost**
All Season Solar Cooker	1	$95	$95
Raspberry Pi	1	$40	$40
SD card	1	$12	$12
Edimax WiFi Dongle	1	$8	$8
MAX31855	1	$15	$15
Photoresistors	4	$0.50	$2
Gearmotor	1	$25	$25
H-bridge	1	$2	$2
4.5V 80mA solar cells	2	$4	$8
Lithium Ion Battery 3.7v 2500mAh	1	$15	$15

PowerBoost 500 Basic - 5V USB Boost	1	$10	$10
Solar Lithium Ion charger	1	$17.50	$17.50
6V 2W Solar panel	1	$25	$25
Total Cost			$274.50

Results

The final oven design had autonomous optimal angle orientating abilities as well as temperature reporting and command input from the user via Gmail. Unfortunately, the solar battery charging rate did surpass the rate of consumption. This caused the Raspberry Pi to consistently power down after roughly two hours. As most solar cooking sessions require more time than this, meaningful data on the positive effects of the autonomous optimal angle orienting feature could not be determined.

In order to generate enough power for the Pi, a more powerful solar panel could be used. For future iterations of this project, however, I suggest replacing the Raspberry Pi with a low power microcontroller. The job of the Pi could easily be accomplished by another device using less energy.

For future development on this project, I also suggest moving away from communication through Gmail. Most email clients have protection against bots operating an account. While it is very possible to work around this, other methods will allow more control. This is especially important to consider if the users communication is ever to be packaged in a phone app.

Another goal for future development is to add an extra degree of rotation to the system. As it is, the oven rotates about the Y-axis only. The oven itself is meant for adjustments in the vertical axis as well. Adding this extra rotation would require a more complicated motor interface, but would certainly increase cooking speeds. Additional commands to find and orient towards the worst cooking angle would also be useful to stop cooking when the food is done.

Conclusion

This project has served to test several initially design ideas of a smart solar oven. The motor, motor control, temperature reading, and optimum cooking angle detection systems worked well. Methods of communication with the user should be revisited to allow more options in the future. Lastly, power generation did not meet requirements.

Though the final oven design was not able to be used to collect data for analysis of its effectiveness, those successful systems used here, as well as an approximate cost, stand as a useful resource and teaching tool for those wishing to adapt solar ovens in the future.

Appendix

checkGmail.py

```
import os
import time
import threading
import feedparser
import smtplib
import imaplib
import RPi.GPIO as GPIO
import Adafruit_GPIO.SPI as SPI
import Adafruit_MAX31855.MAX31855 as MAX31855

from threading import Timer
from email.mime.text import MIMEText

MAILTO = "8582139276@messaging.sprintpcs.com"
USERNAME = "graggpi@gmail.com"
PASSWORD = "" #Not provided

GPIO.setmode(GPIO.BOARD)
GPIO.setup(3, GPIO.IN)

# Define a function to convert celsius to fahrenheit.
def c_to_f(c):
    return c * 9.0 / 5.0 + 32.0

# Raspberry Pi software SPI configuration.
CLK = 25
CS = 24
DO = 18
sensor = MAX31855.MAX31855(CLK, CS, DO)

def MarkAsRead():
  "Marks all emails as being read"
  obj = imaplib.IMAP4_SSL('imap.gmail.com', '993')
  obj.login(USERNAME, PASSWORD)
  obj.select('Inbox')
  typ ,data = obj.search(None,'UnSeen')
  obj.store(data[0].replace(' ',','),'+FLAGS','\Seen')
  return

def Repeat_phone(interval):
        while True:
        os.system("sudo python /home/pi/Oven/textTemp.py")
                time.sleep(interval)

def Check_mail():
        response = feedparser.parse("https://" + USERNAME + ":" + PASSWORD + "@mail.google.com/gmail/feed/atom")
        unread_count = int(response["feed"]["fullcount"])

        print 'Checking mail...'
        if unread_count > 0:
                for i in range(0,unread_count):
                        print "(" + str((i+1)) + "/" + str(unread_count) + ") " + response['items'][i].title
                        command = response['items'][i].title
                        ndx = find(command, ':')
                        commandNum = 0
                        if ndx != -1:
                                command = command[:ndx]
                                commandNum = command[ndx:]
                        print command
                        print commandNum
                        if response['items'][i].title == 'REPORT':
```

```
                                os.system("sudo python /home/pi/Oven/textTemp.py")

                        elif response['items'][i].title == 'REPEAT':
                                interval = 60 # TODO: Read input from response line
                                t = threading.Thread(target=Repeat_phone, args=(interval,))
                                # threads.append(t)
                                t.start()

                        elif response['items'][i].title == 'ANGLE':
                                os.system("sudo python /home/pi/Oven/findAngleGM.py")

                        elif response['items'][i].title == 'CLEAR':
                                t.cancel() # TODO: not stop thread

                MarkAsRead()
        else:
                print 'No new messages'

def Loop_check(interval):
        Check_mail()
        args = [interval]
        Timer(interval, Loop_check, args).start()

Loop_check(60)
GPIO.cleanup()
```

textTemp.py

```
import feedparser
import smtplib
import imaplib
import RPi.GPIO as GPIO
import Adafruit_GPIO.SPI as SPI
import Adafruit_MAX31855.MAX31855 as MAX31855

from email.mime.text import MIMEText

MAILTO = "8582139276@messaging.sprintpcs.com"
USERNAME = "graggpi@gmail.com"
PASSWORD = "" #Not provided


# Define a function to convert celsius to fahrenheit.
def c_to_f(c):
    return c * 9.0 / 5.0 + 32.0

# Raspberry Pi software SPI configuration.
CLK = 25
CS = 24
DO = 18
sensor = MAX31855.MAX31855(CLK, CS, DO)

def Report_phone():
  print 'Generating Report...'
  temp = sensor.readTempC()
  report = 'Thermocouple Temperature: {0:0.3F}*C / {1:0.3F}*F'.format(temp, c_to_f(temp))
  print 'Sending Report to Phone...'
  msg = MIMEText(report);
  msg['Subject'] = 'GraggPi Message'
  msg['From'] = USERNAME
  msg['To'] = MAILTO

  server = smtplib.SMTP('smtp.gmail.com:587')
  server.ehlo_or_helo_if_needed()
  server.starttls()
```

```
  server.ehlo_or_helo_if_needed()
  server.login(USERNAME,PASSWORD)
  server.sendmail(USERNAME, MAILTO, msg.as_string())
  server.quit()
  return

Report_phone()
temp = sensor.readTempC()
report = 'Thermocouple Temperature: {0:0.3F}*C / {1:0.3F}*F'.format(temp, c_to_f(temp))
print report

GPIO.cleanup()
```

findAngleGM.py

```
import RPi.GPIO as GPIO, time, os

def move(pin1, pin2, dir):
        if dir == 1:
                GPIO.output(pin1, GPIO.HIGH)
                GPIO.output(pin2, GPIO.LOW)
        else:
                GPIO.output(pin2, GPIO.HIGH)
                GPIO.output(pin1, GPIO.LOW)

        time.sleep(1.5)
        GPIO.output(pin1, GPIO.LOW)
        GPIO.output(pin2, GPIO.LOW)


def RCtime (RCpin):
        reading = 0
        GPIO.setup(RCpin, GPIO.OUT)
        GPIO.output(RCpin, GPIO.LOW)
        time.sleep(0.1)
        GPIO.setup(RCpin, GPIO.IN)
        # This takes about 1 millisecond per loop cycle
        while (GPIO.input(RCpin) == GPIO.LOW):
                reading += 1
        print reading
        return reading

movePin1 = 15
movePin2 = 13
readPin = 7

CW = 1
CCW = 0

GPIO.setmode(GPIO.BOARD)
GPIO.setup(movePin1, GPIO.OUT)
GPIO.setup(movePin2, GPIO.OUT)

dir = CW
lastMoves = [0, 0]
while (lastMoves[0] + lastMoves[1] != 3) : # stop when you switch direction
        lastMoves[1] = lastMoves[0]
        solarB = RCtime(readPin)
        move(movePin1, movePin2, dir)
        solarA = RCtime(readPin)
        if solarA < solarB: # lower solar reading is better
                lastMoves[0] = 1
                dir = CW
        else:
                lastMoves[0] = 2
```

```
                    dir = CCW
            time.sleep(0.5)

GPIO.cleanup()
```