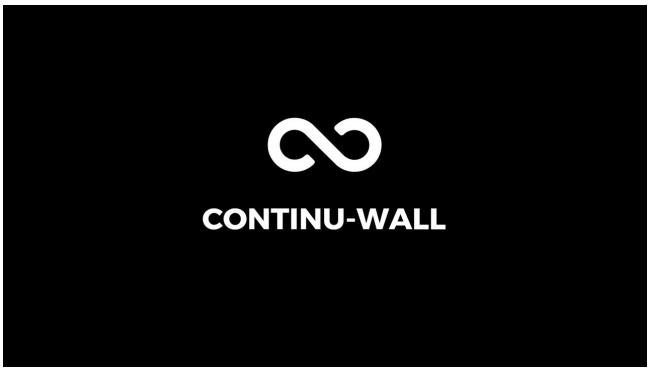
Continu-wall: A creative surface to connect children.

Michael Xiao
Cornell University
Ithaca, NY
mfx2@cornell.edu

Matthew Daniel
Cornell University
Ithaca, NY
mrd89@cornell.edu



ABSTRACT

The Continu-wall is an in-home product designed for children that has two main objectives:

1. To enable creativity through a large continuous physical canvas and
2. To connect people across different cultures and large distances

The device contains a continuously scrolling wallpaper that allows the canvas to be moved into and out of each side of the wall. This allows children to physically save, reset, and revisit drawings while providing a large, expansive canvas that allows them to spatially experience their art - a critical concept for their creativity, especially at such a young age.

Additionally, Continu-wall houses a camera, touch screen GUI, and plotting system. These additional features allow the user to save their own drawings and send them over Wifi to those around the globe. The plotting mechanism can be used to recreate friends' or strangers' drawings from pictures sent to the device. This digital communication allows children from different locations to connect through the universal language of art and collaborate and share their drawings. Figure 1 shows the front view of Continu-wall.

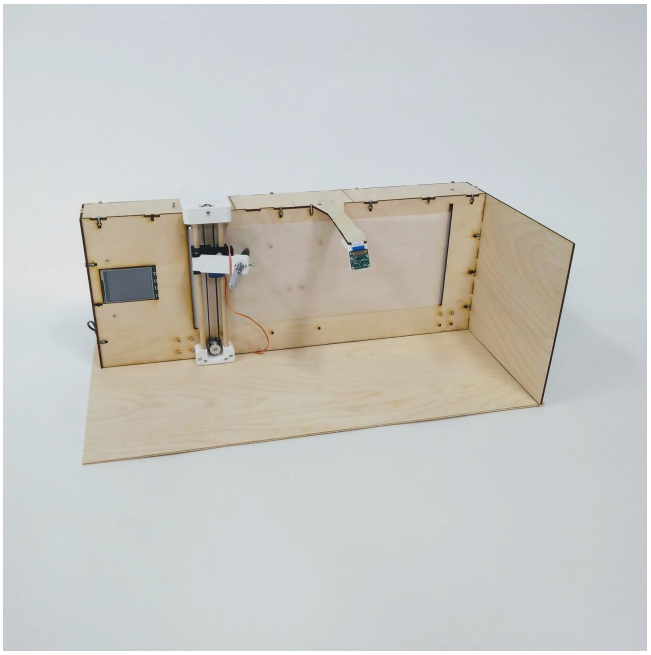


Figure 1: Continu-wall

SCENARIO

Samuel and Lizzie are two six year olds living in the suburbs of New Jersey that love to draw. On weekends, they would often have play dates in which they worked together on the same canvas to learn and draw together, building a bond as well. Unfortunately, Lizzie recently moved across the country to California and since, Samuel has become a bit antsy without an art partner. While home alone on the weekends, Samuel has started drawing on the walls of his house as well leading to frustration from his parents.

Recognizing Samuel's passion and talent with drawing, his parents decide to install the Continu-wall in the house, encouraging him to draw on it. With the device, Samuel comes home

every day to draw new things on large surface and he loves to compare his drawings to his past work saved on the wall's scroll. Furthermore, Samuel has been to stay in touch with Lizzie by sending his drawings to her through the Continu-wall. On Monday Samuel draws a square and sends it to Lizzie. Lizzie receives the drawing and sends back a house that was drawn from the square. Samuel proceeds to finish the drawing by adding trees, people, and some windows to the house.

In addition to staying in touch with Lizzie, Samuel sends his art to his pen pal in France who loves to draw. Even though there is a language barrier, the two are able to fully appreciate the universal language of art through a collaborative work. Through this experience, Samuel is able to appreciate other cultures and get in touch with those outside his immediate bubble.

OPERATION

In its current state, the prototype of Continu-wall is driven by 3 different stepper motors; two that control the scrolling and one for controlling the vertical plotting motion. One small servo is also used to actuate the pen. These actuators are controlled by a Raspberry Pi 3. The Raspberry Pi is also outfitted with a camera that sits on the top of the frame and a touch screen display with four physical buttons as well. Due to our limited time frame, we chose to forego the touch screen capability to focus on other aspects of our projects as the display proved to be rather challenging to set up. As such, the four buttons serve as the user input. Within the touch screen, we utilized the PyGame library to create a GUI and different menu levels. In its initialization state, the display shows the logo, the current menu level, and the operation of the four buttons on the side. This is shown in figure 2.

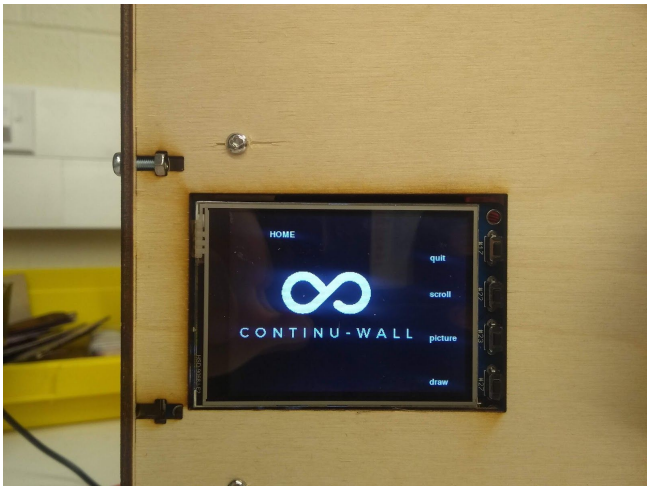


Figure 2: Initial screen

As seen the buttons allow the user to navigate to other menu levels. The quit button exits the program for use with other programs (or in our case to reprogram the script). The scroll button brings the user to the scrolling menu that allows the user to move the screen back and forth. The picture button actuates the camera and snaps a picture of the screen. The user is shown a preview of this capture before getting options to either save it on the device locally, send it via email, share the picture with another Continu-wall device, or retake the picture. The email functionality can be seen in figure 3 below.

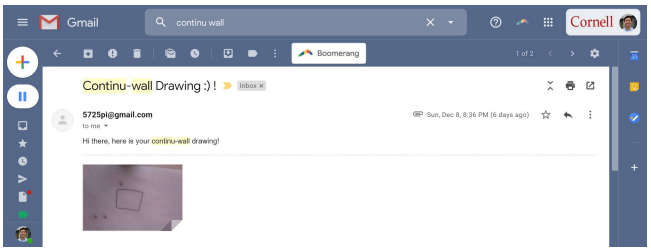


Figure 3: Sample email from Continu-wall

Finally, there is the draw button which brings the user to the drawing menu. In this menu, the user can tell the plotting system to draw primitive shapes to start some artwork off of or download an image from another user. The image can then be processed by noise filtering, thresholding, Sobel's edge detection, and vectorization to generate GCODE to tell the plotting steppers and pen servo how to respond in order to draw the

desired picture. All menus have a back button that brings the user back to the main menu.

LIST OF COMPONENTS

Electrical Components

- Raspberry Pi 3 x1
- Touchscreen TFT display x1
- Raspberry Pi Breakout board x1
- Nema 17 stepper motor x3
- non-continuous micro servo x1
- PiCamera module x1
- DRV8825 Pololu stepper motor driver x2
- 12V 3A power supply x1
- UA7805 5V voltage step down x1

Hardware Components

- Assorted M3 hardware
- M3 heat inserts
- 8mm wooden dowels
- Timing pulleys x2
- Timing belt
- 6" paper roll

PROCESS OF CONSTRUCTION

This project began by deciding the scale at which we wanted to build the prototype. Since the full scale will be 3 feet and around 6 feet long, it is not feasible for us to make a prototype at that scale. Rather, we aimed for a ⅛ scale model. As a result, our prototype measures about 8" tall, 3" deep, and 22" long. This model is small enough to be feasible to build but large enough to convey the intent of the model.

After deciding the scale, we started designing a laser cut frame of that size that serves as both the backboard for the paper but also the general structure of the project. The frame consists of ⅛" plywood joined together with M3 hardware through the use of laser cut T-slots. The initial rendering of this frame is shown in figure 4. As seen, there are different attachment points for the three stepper motors, a window for the touchscreen display, two vertical slots for the paper scroll to move through. The figure also shows the various 3D printed pieces which will be discussed further in the paper.

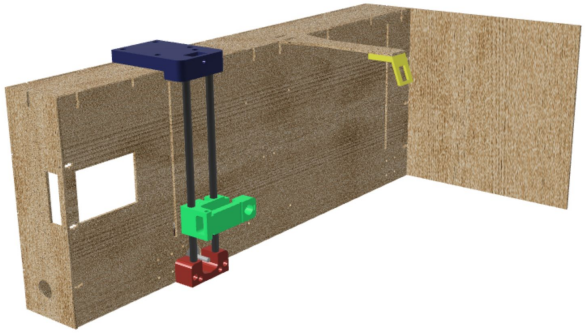


Figure 4: Rendering of frame, 3D printed pieces highlighted in colors

With the frame complete, we designed a 3D printed bracket for each stepper motor which connects directly to the frame. These brackets make use of heat inserts to hold everything steadily. To the rightmost stepper, we designed and printed a roll to attach the paper to in order to keep tension in the paper when moving back and forth. To the leftmost stepper we designed an attachment to snugly hold the roll from both sides. Each of these motors are controlled by the Raspberry PI with stepper motor drivers to control them. The two bottom stepper motors are routed to the same motor driver such that they will always stay synced from a hardware perspective. Additionally, we utilize an external 12V power supply to power each motor controller due to the high current draw from the stepper motors. With all motors incorporated, we were able to move the paper back and forth with ease. Figure 5 shows these stepper holders.

Figure 5: Stepper holder

With the scrolling mechanism complete, we put our attention to the drawing mechanism which houses a pen or marker and can move vertically and lift the pen off the paper. The pen holder uses a simple servo motor and rubber band to control if the pen is up against the paper writing or pulled away. When the servo motor is twisted, the arm will push the pen away from the wall. When the arm is down, the rubber band holds the pen snuggly to the paper. The pen holder is attached to two wooden linear rails to keep things aligned and a belt drive which is powered by another Nema 17 stepper motor routed through a second motor driver. These two rails are held in place by two 3D printed mounts which attach directly to the top and bottom of the Continu-wall's frame. Figure 6 shows the pen holder and carriage parts.

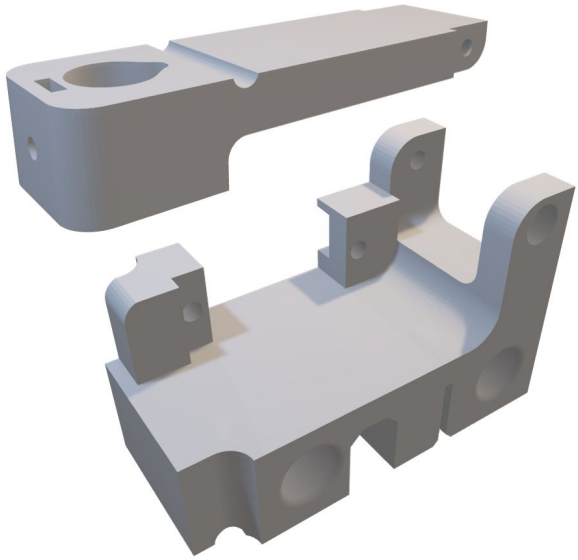


Figure 6: Pen holder and vertical carriage

Lastly, we mounted the touch screen, Pi, and camera to the frame. The touch screen fits over the GPIO pins on the Pi and includes a breakout cable to retain access to other pins. The Pi and screen is held in place by a small bracket that screws into the front face. The camera sits on a thin laser cut arm that juts out from the main frame. A 3D printed mount holds it at an angle downwards such that it is capturing the main portion of the wall.

Finally, we wired up each of the hardware components onto a breadboard which fits nicely in the back of the device and placed the back cover on to improve the overall look of the product. The overall mounting and fit of the project can be seen in figure 7. As seen, there are ports on the left side of the system that allow power cord to enter the internal and to plug in a mouse and keyboard to make code edits on the Raspberry Pi.

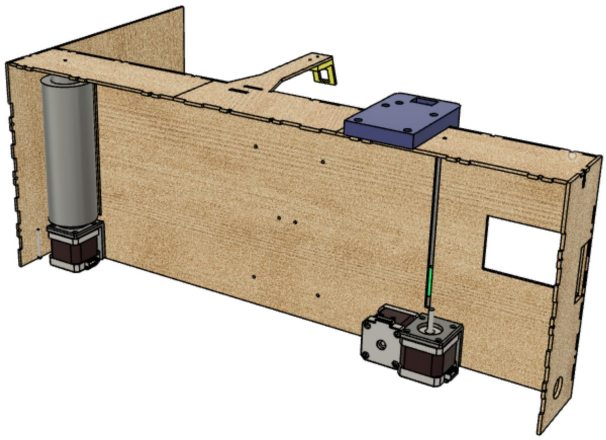


Figure 7: Back view and mounting points

DISCUSSION & FUTURE WORK

This project was a success in our eyes with fewer challenges than we originally expected. W did have some trouble fine tuning our plotting mechanism to move smoothly and actuate the pen correctly. Furthermore, another challenge was having enough time to implement the camera and GCODE experience we had planned to complete. The image processing proved to be difficult and we did not have the resources to fully implement making the transition from a vectorized image to GCODE commands.

Following the success of this prototype, there are numerous objectives to complete with the current scale before creating a full-scale prototype. As mentioned earlier, we did not have time to include a GCODE interpreter for drawing images and this is the next step in this prototype. With this implemented, a user could input any GCODE image and have our system draw it onto the paper

roll. Additionally, we wish to see this system to include more social aspects than currently implemented. A future design will include a feature in which users can create “groups” where each person can upload and share their drawing in addition to downloading their friends’ artwork. The GUI could also be cleaned up to look more user friendly and integrated with full touch screen capability rather than the four physical buttons currently in use. This social aspect is slightly lacking currently in our design and we hope that in the future to make sure children can stay connected from anywhere in the world.

One of the biggest challenges that Coninu-wall faces when moving to the full-size prototype is keeping the cost down while making sure the functionality is kept the same or improved.

VIDEO

A video of describing our product and its functions can be found here:

https://youtu.be/OEXYxr9qIEA

CODE

Our code was written in python on the Raspberry Pi with use of the GPIO, PiCamera, and PyGame libraries. The full code listings can be found below and within our project folder here: https://drive.google.com/drive/u/0/folders/1B7AoQNm9wkt6KTSjg85iz7M8SIg6vykr

main.py

```
#
# Michael Xiao (mfx2) and Matthew Daniel
(mrd89)
# Continu-wall
#

# import libraries
import RPi.GPIO as GPIO
import time
import os
from picamera import PiCamera
import numpy as np
import sys, pygame
from PIL import Image

import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
from email.mime.base import MIMEBase
from email import encoders


os.putenv('SDL_VIDEODRIVER', 'fbcon')   #
Display on piTFT
os.putenv('SDL_FBDEV', '/dev/fb1')     #
os.putenv('SDL_MOUSEDRV', 'TSLIB')      #
Track mouse clicks on piTFT
os.putenv('SDL_MOUSEDEV',
'/dev/input/touchscreen')

pygame.init()
pygame.mouse.set_visible(False)

# set up screen
size = width, height = 320, 240
black = 0,0,0
white = 255, 255, 255
red = 255,0,0
green = 0,255,0
radius = 50

my_font= pygame.font.Font(None, 16)
screen = pygame.display.set_mode(size)
screen.fill(black)

GPIO.setmode(GPIO.BCM)

GPIO.setup(17, GPIO.IN,
pull_up_down=GPIO.PUD_UP)
GPIO.setup(22, GPIO.IN,
pull_up_down=GPIO.PUD_UP)
GPIO.setup(23, GPIO.IN,
pull_up_down=GPIO.PUD_UP)
GPIO.setup(27, GPIO.IN,
pull_up_down=GPIO.PUD_UP)

camera = PiCamera()


time_start = time.time()

state = 0
```

```
SERVO = 6

#
https://www.rototron.info/raspberry-pi-stepper
-motor-tutorial/
DIR = 13   # Direction GPIO Pin
STEP = 16  # Step GPIO Pin

DIR2 = 5
STEP2 = 12

CW = 1     # Clockwise Rotation
CCW = 0    # Counterclockwise Rotation
SPR = 200   # Steps per Revolution

GPIO.setup(DIR, GPIO.OUT)
GPIO.setup(STEP, GPIO.OUT)
GPIO.setup(DIR2, GPIO.OUT)
GPIO.setup(STEP2, GPIO.OUT)

GPIO.output(DIR, CW)
GPIO.output(DIR2, CW)

# setup for servo
GPIO.setup(SERVO, GPIO.OUT)
servo = GPIO.PWM(SERVO, 46.5)

delay = .0208

# send an email
def send_image():

        email_user = '5725pi@gmail.com'
        email_password = 'gnarly!1'
        email_send = 'mfx2@cornell.edu'

        subject = 'Continu-wall Drawing :) !'
        msg = MIMEMultipart()
        msg['From'] = email_user
        msg['To'] = email_send
        msg['Subject'] = subject

        body = 'Hi there, here is your
continu-wall drawing!'
        msg.attach(MIMEText(body,'plain'))

        filename='/home/pi/Downloads/image.png'
        attachment  =open(filename,'rb')

        part =
```

```
MIMEBase('application','octet-stream')
        part.set_payload((attachment).read())
        encoders.encode_base64(part)

part.add_header('Content-Disposition',"attachm
ent; filename= "+filename)

        msg.attach(part)
        text = msg.as_string()
        server =
smtplib.SMTP('smtp.gmail.com',587)
        server.starttls()
        server.login(email_user,email_password)



server.sendmail(email_user,email_send,text)
        server.quit()


#controlling steps for scrolling
def step(step_count):
        if step_count < 0:
                GPIO.output(DIR, CCW)
        else:
                GPIO.output(DIR, CW)
        for x in range(step_count):
            GPIO.output(STEP, GPIO.HIGH)
            time.sleep(delay)
            GPIO.output(STEP, GPIO.LOW)
            time.sleep(delay)

#controlling steps for pen
def step2(step_count):
        if step_count < 0:
                GPIO.output(DIR2, CCW)
        else:
                GPIO.output(DIR2, CW)
        for x in range(step_count):
            GPIO.output(STEP2, GPIO.HIGH)
            time.sleep(delay)
            GPIO.output(STEP2, GPIO.LOW)
            time.sleep(delay)

# making a button
def make_button(text,location):
    text = my_font.render(text,True,white)
    text_rect =
text.get_rect(topleft=location)
    return text,text_rect
```

```
# making the sidebar
def make_sidebar(text1, text2, text3, text4,
menu):
	left_1, left_1_rect =
make_button(text1, (270,50))
	screen.blit(left_1, left_1_rect)
	left_1, left_1_rect =
make_button(text2, (270,100))
	screen.blit(left_1, left_1_rect)
	left_1, left_1_rect =
make_button(text3, (270,160))
	screen.blit(left_1, left_1_rect)
	left_1, left_1_rect =
make_button(text4, (270,220))
	screen.blit(left_1, left_1_rect)
	left_1, left_1_rect = make_button(menu,
(50,20))
	screen.blit(left_1, left_1_rect)
	pygame.display.flip()

oldState = 0

logo =
pygame.image.load('/home/pi/Downloads/logo.png
')

while True:

	if state != oldState:
		screen.fill(black)
		time.sleep(1)
	oldState = state
	time.sleep(0.1)
	print(state)
	screen.fill(black)


	if state == 0: # init state

		screen.blit(logo, (-3,40))
		make_sidebar("quit", "scroll",
"picture" , "draw", "HOME")

		if (not GPIO.input(22)):
			print("scroll mode")
			state = 3
		elif (not GPIO.input(23)):
			print(" take a picture")
```

```
camera.capture('/home/pi/Downloads/image.png')
			im =
Image.open("/home/pi/Downloads/image.png")
			im = im.rotate(180)

im.save("/home/pi/Downloads/image.png")

			img =
pygame.image.load('/home/pi/Downloads/image.pn
g')
			img =
pygame.transform.scale(img, (256, 144))

			state = 2
		elif (not GPIO.input(27)):
			print(" draw mode")
			state = 1
		elif (not GPIO.input(17)):
			print("quit")
			break

	elif state == 1: #draw mode

		screen.blit(logo, (-3,40))
		make_sidebar("back", "square",
"circle" , "load", "DRAW")

		if (not GPIO.input(22)):
			print("draw a square")
			os.system("python
/home/pi/Downloads/servo2.py")
			os.system("python
/home/pi/Downloads/stepper2.py 100")
			os.system("python
/home/pi/Downloads/stepper.py 400")
			os.system("python
/home/pi/Downloads/stepper2.py -150")
			os.system("python
/home/pi/Downloads/stepper.py -400")


			os.system("python
/home/pi/Downloads/servo.py")
			# draw square

		elif (not GPIO.input(23)):
			print("draw a circle")
			# draw circle

		elif (not GPIO.input(27)):
```

```
                        print(" load")
                        # draw triangle

                elif (not GPIO.input(17)):
                        print("back")
                        state = 0

        elif state == 2: #picture mode
                # display the picture
                screen.blit(img, (-3,40))
                make_sidebar("back", "send",
"share" , "", "PICTURE")

                if (not GPIO.input(22)):
                        print("send")
                        state = 0
                elif (not GPIO.input(23)):
                        print("share")
                        state = 0
                elif (not GPIO.input(17)):
                        print("back")
                        state = 0

        elif state == 3: #scroll mode
                screen.blit(logo, (-3,40))
                make_sidebar("back", "left",
"right" , "", "SCROLL")

                if (not GPIO.input(22)):
                        #os.system("python
/home/pi/Downloads/servo.py")
                        print("scroll forward")
                        os.system("python
/home/pi/Downloads/stepper2.py 30")
                elif (not GPIO.input(23)):
                        #os.system("python
/home/pi/Downloads/servo2.py")
                        print(" scroll back")
                        os.system("python
/home/pi/Downloads/stepper2.py -30")
                elif (not GPIO.input(17)):
                        print("back")
                        state = 0


GPIO.cleanup()
```

stepper.py

```
import sys
import os
import RPi.GPIO as GPIO
import numpy as np
import time
from time import sleep

steps = int(sys.argv[1])
GPIO.setmode(GPIO.BCM)


DIR2 = 5
STEP2 = 12
DIR = DIR2
STEP = STEP2

CW = 1     # Clockwise Rotation
CCW = 0    # Counterclockwise
Rotation
SPR = 200   # Steps per Revolution

GPIO.setup(DIR2, GPIO.OUT)
GPIO.setup(STEP2, GPIO.OUT)

delay = 0.00208
step_count=steps

if (steps < 0):
  GPIO.output(DIR, CW)
else:
  GPIO.output(DIR,CCW)

for x in range(abs(step_count)):
  GPIO.output(STEP, GPIO.HIGH)
  sleep(delay)
  GPIO.output(STEP, GPIO.LOW)
  sleep(delay)
```

servo.py

```
#!/usr/bin/env python3

import sys
import os
import RPi.GPIO as GPIO
import numpy as np
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(6, GPIO.OUT)

# Start servo close to stopped
# Create a PWM instance:

time_on_stop = 1.505 # ms
time_on_cw = 1.3 # ms
time_on_ccw = 1.7 # ms
time_off = 20 # ms

maxDC = (time_on_ccw/(time_on_ccw
+ time_off))*100
minDC = (time_on_cw/(time_on_cw +
time_off))*100
maxFreq = 1/(0.001*(time_on_ccw +
time_off))
minFreq = 1/(0.001*(time_on_cw +
time_off))
stoppedDC =
(time_on_stop/(time_on_stop +
time_off))*100
stoppedFreq =
1/(0.001*(time_on_stop +
time_off))

p = GPIO.PWM(6, stoppedFreq)
p.start(stoppedDC)
```

```
p.ChangeFrequency(maxFreq) # where
freq is the new frequency in Hz
p.ChangeDutyCycle(maxDC) # where
0.0 <= dc <= 100.0
time.sleep(1)

p.stop()
```

mail.py

```
import smtplib
from email.mime.text import
MIMEText
from email.mime.multipart import
MIMEMultipart
from email.mime.base import
MIMEBase
from email import encoders

x = 1

if (x ==1):

      email_user =
'5725pi@gmail.com'
      email_password = 'gnarly!1'
      email_send =
'mfx2@cornell.edu'

      subject = 'Continu-wall
Drawing :) !'
      msg = MIMEMultipart()
      msg['From'] = email_user
      msg['To'] = email_send
      msg['Subject'] = subject

      body = 'Hi there, here is
```

```
your continu-wall drawing!'

msg.attach(MIMEText(body,'plain'))


filename='/home/pi/Downloads/image
.png'
      attachment
=open(filename,'rb')

      part =
MIMEBase('application','octet-stre
am')

part.set_payload((attachment).read
())
      encoders.encode_base64(part)

part.add_header('Content-Dispositi
on',"attachment; filename=
"+filename)

      msg.attach(part)
      text = msg.as_string()
      server =
smtplib.SMTP('smtp.gmail.com',587)
      server.starttls()

server.login(email_user,email_pass
word)



server.sendmail(email_user,email_s
end,text)
      server.quit()
```