EEC 172 Final Project – Ultrasound-Based Virtual Sketchpad

Kolin Guo, Yizhi Tao June 21, 2018

Introduction

Today many artists use digital sketchpads to draw artworks on the computer. However, the physical sketchpad often takes up much space and requires a stylus. We want to design a "virtual" sketchpad that uses sensors to detect the movement of the pen, hence eliminating the need of a physical sketchpad. In this open lab project, we implement a "virtual" sketch pad of 32 by 32 pixels using TI CC3200 LaunchPad and two ultrasonic sensors. The painting is displayed on the 128x128 OLED screen in real time. User can also interact with the IR remote to pick the colors of the pen and send the painting to Amazon Web Services (AWS) S3 bucket which automatically converts the painting to a PNG file for downloads.

Top level Description

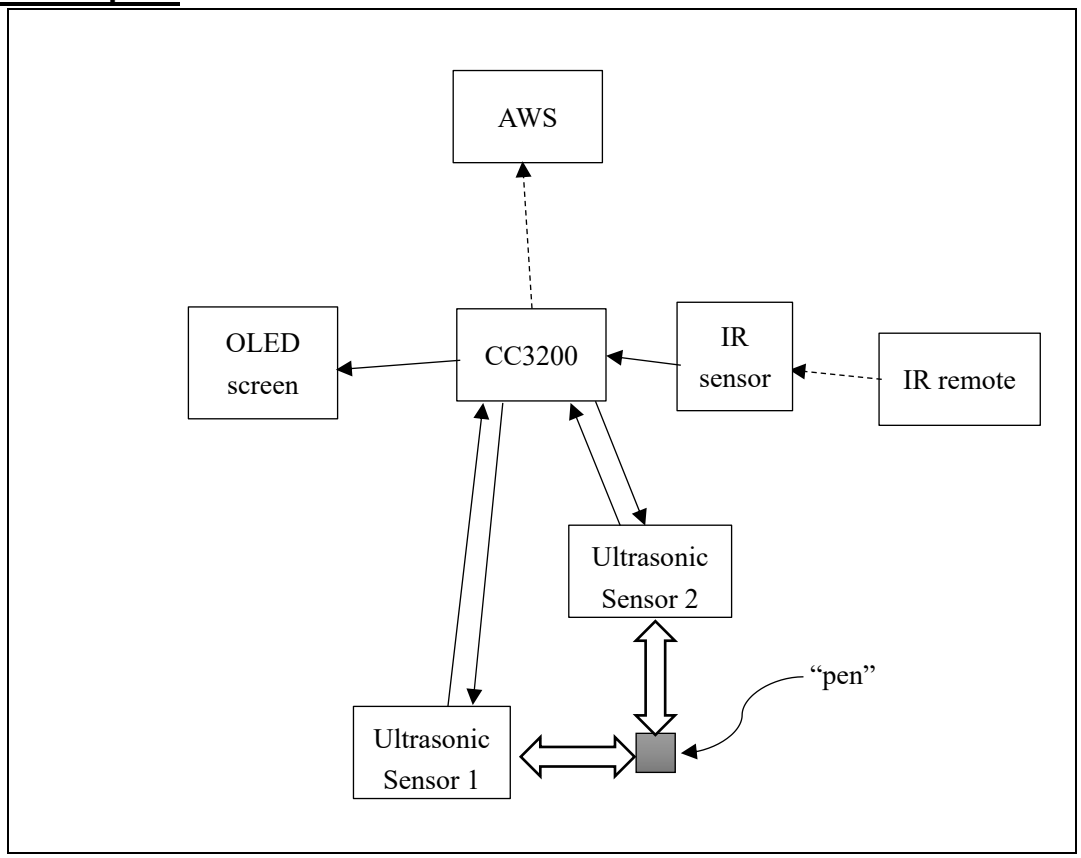

Fig. 1. Top-Level Block Diagram

Fig 1. shows the top-level setup of the system. The CC3200 is connected to two ultrasonic sensors using GPIO pins. It is also connected to an OLED display via SPI protocol to display drawings and an IR sensor via GPIO to receive signals from the IR remote. Finally, the CC3200 is connected to AWS using the built-in Wi-Fi module and RESTful API. Below is a table summarizing the sensor information.

Sensor Name	Descriptions
HC-SR04 Ultrasonic Ranging Module	2cm – 400cm non-contact measurement, 3mm measurement accuracy, 15° measurement angle, 40kHz working frequency, simply connect using GPIO
TSOP1236/31236/31336 IR Receiver Module	36kHz carrier frequency, simply connect using GPIO
AT&T S10-S3 Remote Control	Use NEC code, 12 functional keys
SSD1351 128x128 OLED Display	16-bit RGB565 format, 65536 colors, connect to CC3200 using SPI

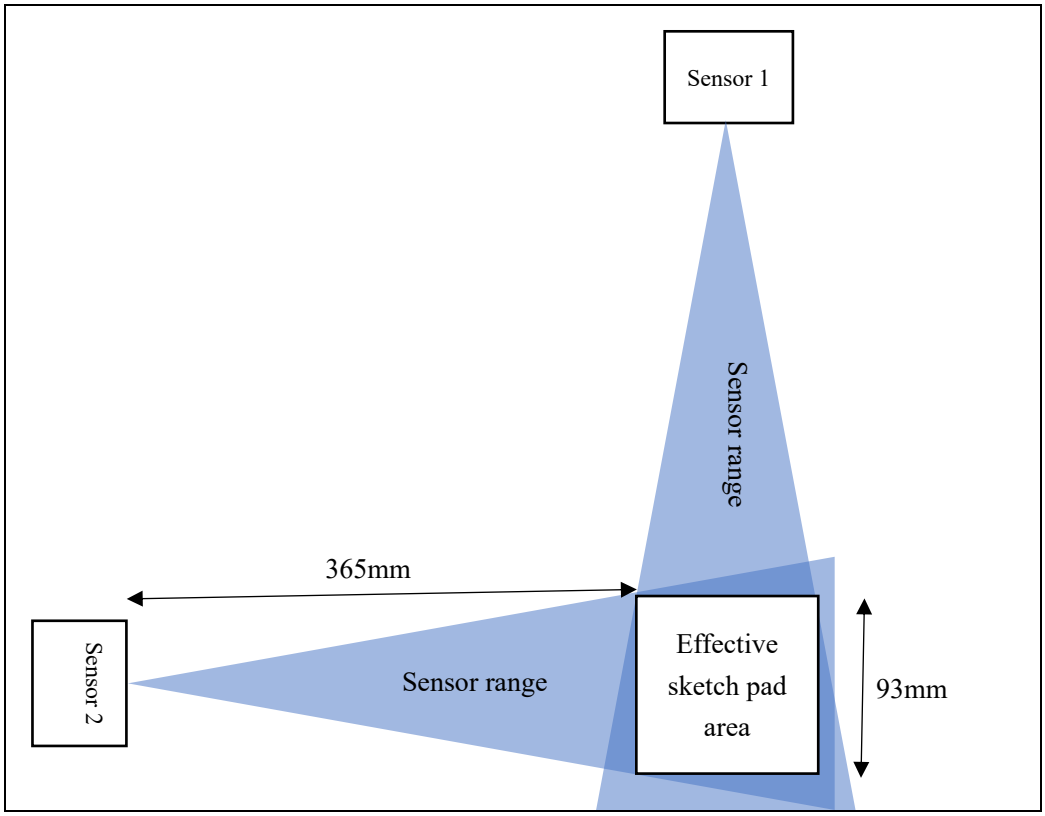

Fig. 2. Top-Down View of the Sketchpad

Fig. 2. shows the top down view of the actual sensor setup. We use two HC-SR04 ultrasonic sensors to measure the distances from the “pen” in perpendicular directions. The pen is a cuboid object in practice because its flat surfaces reflect ultrasonic sound well. The two distances are then converted to x and y coordinates of the pixel on the OLED screen. So, the two sensors can track the movement of the pen and update its position on the OLED screen.

The HC-SR04 ultrasonic sensor has a *trigger* that sends an 8-cycle burst of ultrasonic soundwave at 40kHz at the object. Then the *echo* receives the reflected ultrasonic signal. The distance is calculated using the time interval between trigger and echo: Distance = high-level time * 340m/s /2. The high-level time on the GPIO pin is measured using the general-purpose timers and GPIO interrupts.

HC-SR04 ultrasonic sensors can measure distance from 2cm to 4m with accuracy up to 3mm. It has a measuring angle of 15 degrees. Because of the limited measuring angle, the pen needs to be placed relatively far away (about 4x the sketchpad side) from the sensors to achieve a larger sketch area. After calculations and testing, we decided to have a 32x32 pixels sketchpad on the OLED. With the 3mm accuracy, the length of the side of effective sketch area is calculated to be 31*3mm = 93mm. The distance from one sensor to the nearest side is calculated using trigonometry to be 365mm. Because of limited accuracy and measuring angle, having a higher resolution or a larger sketch area would require larger setup area and we thought it would not be feasible. A "pen" made of LEGO bricks (15mm x 15mm) will be used because its flat surfaces can easily reflect ultrasonic sound.

The buttons (0-9) on the remote are programmed to change the color of the pen according to the color mapping summarized in the table below. The MUTE button is programmed to clear the drawing on the screen and LAST button is programmed to send a string containing the pixel values of the sketch will be sent to AWS.

IR Remote Keys	Color Mapping/Functionality		IR Remote Keys	Color Mapping/Functionality	
0	Black	0x000000	6	Cyan	0x00FFFF
1	White	0xFFFFFF	7	Azure	0x0080FF
2	Red	0xFF0000	8	Blue	0x0000FF
3	Orange	0xFF8000	9	Purple	0x7F00FF
4	Yellow	0xFFFF00	MUTE	Clear current drawing	
5	Green	0x00FF00	LAST	Send current drawing to AWS	

CC3200 Connections

The pin configuration of TI CC3200 LaunchPad is summarized in the table below.

Package Pin #	UART/SPI/GPIO Signals	Usage
64	GPIO_09	Output, ultrasound sensor *trigger* signal
8	GPIO_17	Input, ultrasound sensor #1 *echo* signal
50	GPIO_00	Input, ultrasound sensor #2 *echo* signal
63	GPIO_08	Input, IR sensor signal
61	GPIO_06	Output, OLED D/C signal
62	GPIO_07	Output, OLED Chip Select signal
18	GPIO_28	Output, OLED Reset signal
5	GSPI_CLK	General SPI Clock
7	GSPI_MOSI	General SPI MOSI
55	UART0_TX	UART0 TX Data
57	UART0_RX	UART0 RX Data
2	GPIO_11	Green LED. Not used actually, can be removed

Code Description

`IRIntHandler()` collects input from IR sensor.

`US1IntHandler()` starts timer of the 1st ultrasonic sensor when positive edge is detected on *echo* pin. Then it records the time and stops the timer when negative edge is detected.

`US2IntHandler()` starts timer of the 2nd ultrasonic sensor when positive edge is detected on *echo* pin. Then it records the time and stops the timer when negative edge is detected.

`TimerIntHandler()` handles timer interrupt for the IR sensor to decode the message.

`ChangeRGBColor(unsigned int)` changes the color in a rainbow fashion according to IR sensor input.

`ResetImage()` clears the OLED screen and resets image with MUTE key.

`GetIRInput()` gets input from IR remote.

`CollectUSData()` collects data from the two ultrasonic sensors, converts them to distances and calls `ConvertToPixelLoc()`.

`ConvertToPixelLoc()` converts distance data to pixel location on OLED.

`DrawSizedPixel(unsigned char)` draws a pixel on OLED of specified size (4-by-4 dots by default).

`SaveToRGBArr()` saves the RGB pixel data to `g_imageRGB[]`.

`StartSketching()` starts sketching on the OLED display by calling `CollectUSData()`, `DrawSizedPixel(4)`, `SaveToRGBArr()` and `GetIRInput()` in an infinite loop.

`http_post()` converts `g_imageRGB[]` array to a JSON readable character string with commas separating the RGB values in hexadecimal format. Then it posts the string to an AWS IoT thing.

Amazon Web Services

The AWS flowchart is shown below as Fig. 3.

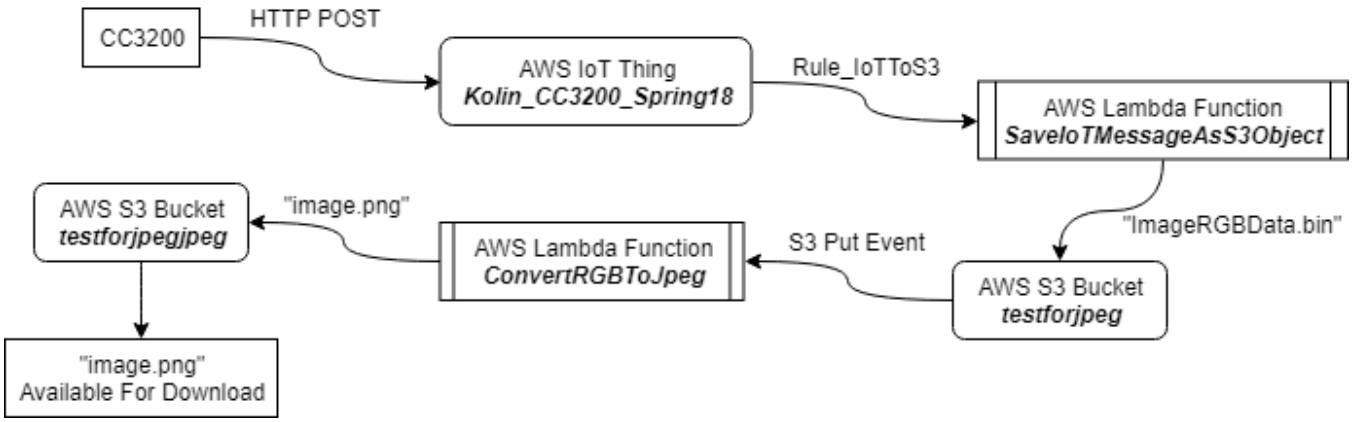

Fig. 3. AWS Flowchart

The drawing is sent to an AWS IoT thing shadow as a string of the RGB values using HTTP POST. When the thing shadow is updated, the ***Rule_IoTToS3*** will invoke a Lambda function called ***SaveIoTMessageAsS3Object*** (Fig. 4), passing the message data. This function strips the RGB data from the IoT message and saves it as a binary file named "***ImageRGBData.bin***" into an S3 bucket (***testforjpeg***). When the binary file is generated in the ***testforjpeg*** S3 bucket, an S3 Put event occurs which invokes another Lambda function called ***ConvertRGBToJpeg*** (Fig. 5), passing the binary file location. This function converts the binary file to a PNG image file and saves it as "***image.png***" into another S3 bucket named ***testforjpegjpeg***. Then, user can download the PNG file. Both Lambda functions are written in Python. ***ConvertRGBToJpeg*** function uses OpenCV and NumPy libraries to convert RGB data to a PNG file.

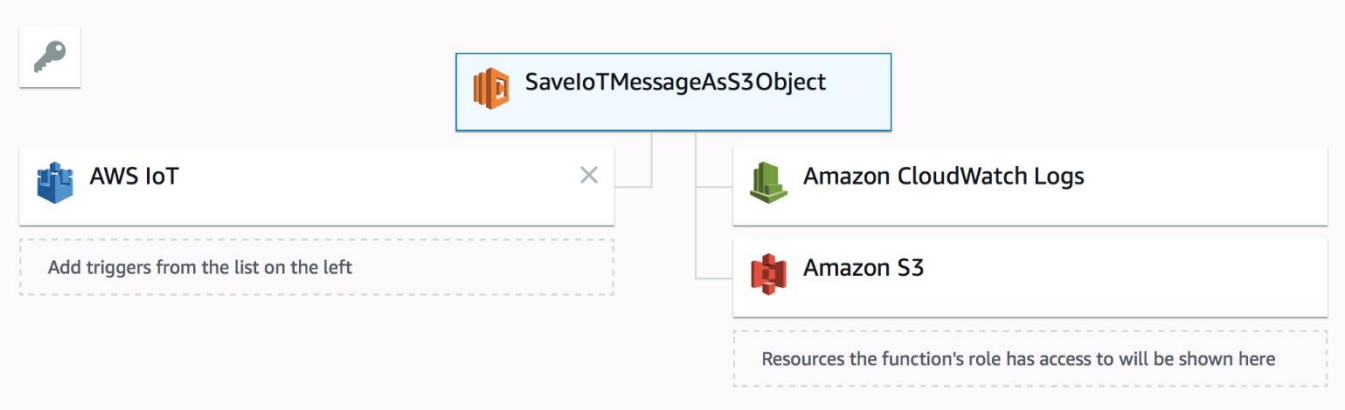

Fig. 4. Lambda Function That Saves IoT Message as Binary File

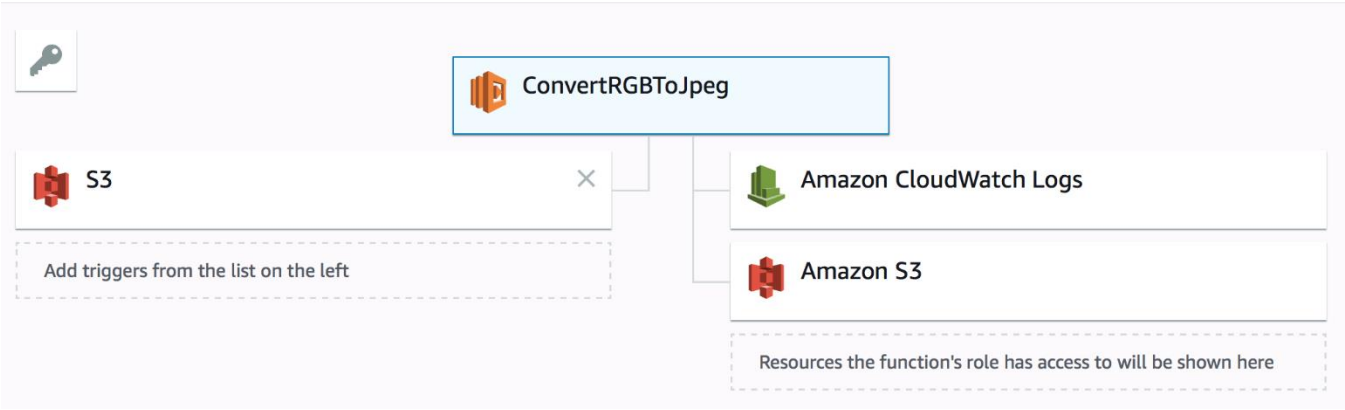

Fig. 5. Lambda Function That Converts Binary RGB to PNG File

Initially, the ***ConvertRGBToJpeg*** function converted the RGB data to a JPEG file. However, we found that our 32x32 image is blurry. This is because JPEG is a lossy compression image format and it does not work very well for an image with such few pixels. Therefore, we modified the ***ConvertRGBToJpeg*** function to save the image as a PNG file which is a lossless image format and thus produces an accurate image.

Difficulties

Due to the limited accuracy and measuring angle of HC-SR04 ultrasonic sensors, we have to come up with solutions in order to measure the location of the pen. For example, the sensors have to the placed far enough to cover the lateral movement of the pen. If the sensor is too close, the pen could easily go out of the measuring angle. If the sensor is too far, implementing such setup can be cumbersome. After calculation, we decided to have an effective sketch area of 93mm x 93mm and place the sensors 365mm away from the closest side of the sketch area.

Converting the RGB array to a string in order to send it using HTTP POST was a challenge. Also, converting the binary to PNG involves many built-in Python libraries that we needed to learn.

Future Improvements

- Better sensors.
  - Because a lot of the design choices were made due to the limitation of the ultrasonic sensors, we want to use better sensors with wider measuring angles and higher accuracy. That way we will have a smaller setup and larger sketch area. We will also be able to draw in higher resolution with more accurate paint brush.
  - Also, the latency of drawing would become significant in professional usage.
- Send file directly to AWS S3 bucket.
  - In this lab, we tried to save the binary file of picture locally on CC3200 flash and send it to AWS S3 bucket using HTTP PUT or POST. We have successfully implemented saving binary file on CC3200 and can open it on a computer. However, in order to directly use HTTP PUT or POST to upload file to S3 bucket, we need to authenticate requests using AWS Signature Version 4. The signature calculation for the authorization header is very complex involving URI encoding, SHA256Hash calculation, and HMAC-SHA256 conversion. As a result, we were not able to implement at this time. In the future, if we can directly send binary files from CC3200 to S3 bucket, we can bypass the size limit of HTTP POST JSON message, which is 8KB, and upload higher resolution images.
- More functionalities.
  - The sketchpad features that we implemented by now are basic color changes (10 different colors) and resetting current drawing. It would be better if we can add more functionalities in the future.
  - Examples are line thickness adjustment, custom color selection, background color selection, zoom in and out, different paint brushes (hardness and feathering) and so on.
- Notify user about successful image generation in AWS S3 bucket.
  - Right now, when the PNG image file is successfully generated in the S3 bucket, there is no notification to the user. In the future, we can send the image through email to user's email address or send a text message to user notifying the results. Another way is to connect to Facebook or Twitter through IFTTT to post the image.
- Allow painting more naturally.
  - Using more sensors, we can allow user to paint by actually holding a pen with hand movement just like painting on paper.
  - However, this is hard to implement with ultrasonic-like distance sensor because there are slight differences in how people holds the pen and draws with it. It's better to either use a trackpad/touchscreen as the sketchpad or add sensors directly to the tip of the pen. However, this deviates from the original idea of this project (a virtual sketchpad) and very few of the finished work can be reused.
- 3D painting with pressure sensing.
  - Add more sensors to detect the pressure user applied to the pen. Use the pressure to adjust the size of the paint brush.
  - In order to implement this functionality, the pressure sensor is better to be attached at the tip of the pen.

Acknowledgement

First and foremost, we would like to show our deepest gratitude to the professor of EEC 172 class in Spring 2018, Professor Soheil Ghiasi, for teaching us knowledge and applications of embedded systems. Then, we would like to appreciate our TA Dan Fong, who helped to create these interesting lab projects and offered us useful advices throughout the quarter. Without him, we could not achieve and learn as much as we did.

Last but not least, we would like to thank another UCD undergraduate student, Sam Minh Truoug, for suggesting us to do JPEG conversion using AWS Lambda function and OpenCV library for Python. Without him, it would take us tremendous amount of time trying to do the conversion locally on CC3200.

References and Resources

For AWS IoT:

1. Creating a Lambda Rule
   https://docs.aws.amazon.com/iot/latest/developerguide/iot-lambda-rule.html

For AWS Lambda Function:

1. Set Up an AWS Account (Administrator) to Invoke and Configure Lambda Functions
   https://docs.aws.amazon.com/lambda/latest/dg/setup.html
2. Set Up the AWS Command Line Interface (AWS CLI)
   https://docs.aws.amazon.com/lambda/latest/dg/setup-awscli.html
3. Sample Events Published by Event Sources (S3 Bucket Put Event)
   https://docs.aws.amazon.com/lambda/latest/dg/eventsources.html#eventsources-s3-put
4. Create a Lambda Deployment Package using an EC2 Instance
   https://docs.aws.amazon.com/lambda/latest/dg/with-s3-example-deployment-pkg.html#with-s3-example-deployment-pkg-python
5. API Reference
   https://docs.aws.amazon.com/lambda/latest/dg/API_Reference.html

For AWS EC2 Instance:

1. Setting Up with Amazon EC2
   https://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/get-set-up-for-amazon-ec2.html

For AWS S3 Bucket:

1. PUT Object using REST API
   https://docs.aws.amazon.com/AmazonS3/latest/API/RESTObjectPUT.html
2. POST Object using REST API
   https://docs.aws.amazon.com/AmazonS3/latest/API/RESTObjectPOST.html
3. Signature Calculations for the Authorization Header: Transferring Payload in a Single Chunk (AWS Signature Version 4) for PUT/GET
   https://docs.aws.amazon.com/AmazonS3/latest/API/sig-v4-header-based-auth.html
4. Signature Calculations for the Authorization Header: Transferring Payload in Multiple Chunks (Chunked Upload) (AWS Signature Version 4) for PUT/GET
   https://docs.aws.amazon.com/AmazonS3/latest/API/sigv4-streaming.html