UC Berkeley, ME100, Spring 2021
Lab 5: Amplitude modulation, rectification and IFTTT

Plan to complete these tasks during the week of March 8
Checkoff due on Gradescope by Monday March 29, 11:59pm

Introduction and objectives

There are two sections in this week's lab:

A. Explore signal rectification, modulation and demodulation

Here, the aims are to:

- Reinforce understanding of the course material on diodes, half- and full-wave rectifiers, amplitude modulation, and RC circuits.
- Be introduced to the concept of digital-to-analog conversion (DAC) and the capabilities and limitations of the ESP32 for DAC.
- Gain practice using the oscilloscope, debugging circuits, and interpreting waveforms.

The modulation schemes used in IoT radio transmission devices are extremely sophisticated compared to the simple AM scheme introduced in lecture and experimented with here, but the classic AM approach provides a good learning model and today's IoT devices build on the same physics.

B. Become familiar with the IFTTT platform for connecting your ESP32 with web services.

So far, you have user MQTT in conjunction with the MQTT and Thingspeak brokers to publish message data via the internet. Now, you will explore a commercial platform called IFTTT (If This Then That) which builds on top of MQTT and is designed to make it simple to connect microcontrollers with web services (e.g. sources of online data). IFTTT also makes it straightforward to use a smartphone app to interface with a microcontroller and control an IoT device. This platform may be very useful for implementing your class project.

Section A: Modulation/demodulation

Step 1: Get DAC outputs working

Download and understand the sample code, ME100_Lab5_sample_AM_code.py, from Files > Labs > Lab 5 on bCourses. This code makes use of the two digital-to-analog pins on the ESP32, each of which allows you to output a controllable voltage ranging from 0 to 3.3 V. We will use this capability to generate an amplitude-modulated sinusoidal wave to test a rectification and demodulation circuit.

With nothing connected to the ESP32 (just plug it into your computer with the USB cable), use `shell49` to run the sample code on your board. Connect the probe and the ground alligator clip of your oscilloscope to GPIO pin 25 (DAC1) and GND respectively (use jump wires to facilitate the

connections). Examine the waveform produced. What is the frequency of the carrier wave? What is the frequency of the modulating signal?

Now connect your oscilloscope between GPIO pin 26 (DAC2) and GND and confirm that there is also a modulated sinusoidal waveform on that pin as well.

To test your rectification circuits, you will need an oscillating voltage signal that *changes sign* periodically. Since any given pin on the ESP32 can output only positive voltages, we will use the difference between the voltages on pins 25 and 26 to provide a sign-changing signal.

From the code, the signal on pin 25 is a sampled version of:

$$x_1(t) = V_s[0.5 + 0.5\sin\omega_c\,(0.5 + 0.5\sin\omega_m)]$$

and the signal on pin 26 is a sampled version of:

$$x_2(t) = V_s(0.5 + 0.5\cos\omega_c\,(0.5 + 0.5\sin\omega_m))$$

where ω_C is the carrier frequency and $\omega_m = \frac{\omega_c}{50}$ is the modulation frequency.

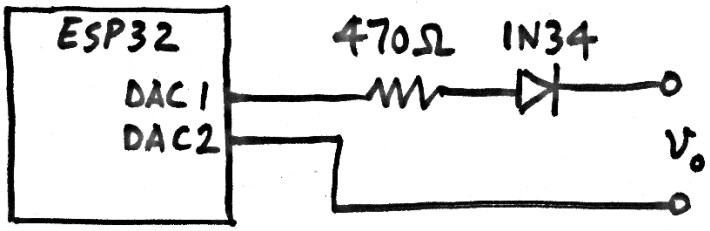
The difference between the voltages is:

$$x_1(t) - x_2(t) = 0.5V_s\,(0.5 + 0.5\sin\omega_m)(\sin\omega_c - \cos\omega_c) = \frac{\sqrt{2}}{2}V_s\,(0.5 + 0.5\sin\omega_m)\cos\left(\omega_c - \frac{3\pi}{4}\right)$$

which is the amplitude-modulated signal we need.

Step 2: Build half-wave rectification circuit

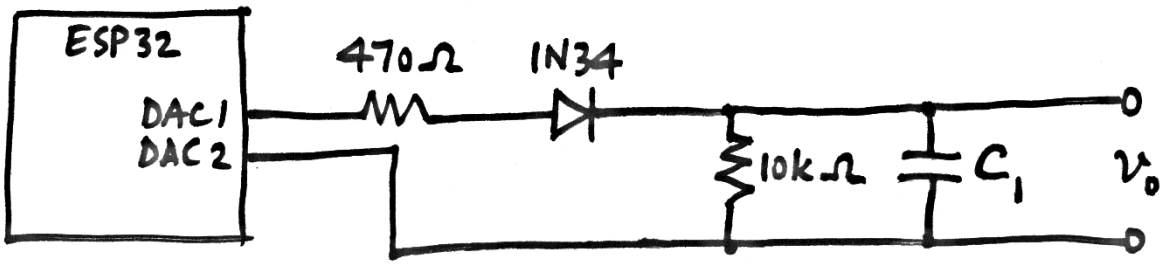
Construct the following circuit on your prototyping board:



The 470 ohm resistor is to ensure that the maximum source or sink current of the ESP32 pins is not exceeded even in case of an accidental short downstream of the rectifier.

Probe the voltage v_o and capture a screenshot from your oscilloscope showing the half-wave-rectified waveform and including about 0.5 to 1 periods of the modulating signal. Use automatic triggering as in Lab 3, and press STOP to freeze the waveform prior to photographing it. (For introductory instructions on oscilloscope usage, refer to the Lab 3 handout or walkthrough video.)

Now connect a resistor of 10 kΩ and a capacitor C_1 across the output terminals:



Select your value of C_1 to filter out the carrier signal as much as possible while retaining the modulating signal. What value of C_1 did you select, and what was your reasoning?

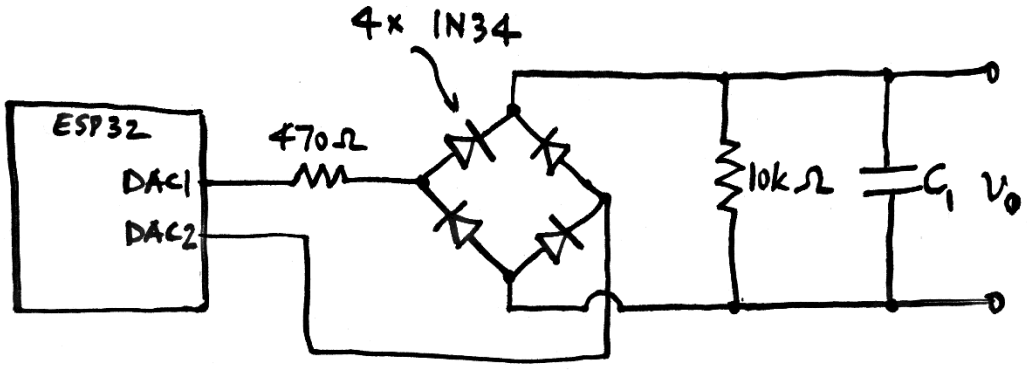
Capture a screenshot of your demodulated signal (200 mV/div and 50 ms/div worked for me). Comment on the shape of the recovered waveform. If you were hoping to recover your original modulating sine signal, would you be satisfied? How would you decide whether you were satisfied?

Now try a much smaller capacitor in place of C_1 (say, 10x smaller). Capture another screenshot of the output signal now. Comment on what you observe.

The sample code modulates the signal 100% — i.e. the amplitude of the transmitted wave falls to zero (or very near zero) periodically. If you used a smaller amount of modulation, could you improve the quality of the recovered waveform?

Step 3: Full-wave rectification circuit

Now implement a full-wave rectifier with four diodes as shown below:



Use the same C_1 as for the half-wave rectifier.

Capture a screenshot of the voltage waveform across the output terminals: (a) with capacitor C_1 in place, and (b) with capacitor C_1 removed. Confirm whether the full-wave rectifier is working as expected

with the capacitor in place. Comment on the quality of the recovered (demodulated) signal compared to the output of the half-wave rectifier.

Devise a way of measuring the voltage drop across the 1N34 diode when it is forward-biased and conducting in this circuit (e.g. by making and interpreting a particular oscilloscope measurement from your circuit). Report your measured voltage drop. Do some online research to check whether your value is reasonable, and comment on your finding in relation to the rule-of-thumb introduced in lectures of a 0.6–0.7 V forward voltage drop for diodes.

Section B: IFTTT

In this section you'll familiarize yourself with two resources that may be very useful for your projects: Adafruit.io and IFTTT. This section is not intended to be particularly taxing and involves mostly following the many small steps so you are aware of the capabilities at your disposal.

Step 1: Connect to internet

Ensure that you are able to connect to the Internet with your ESP32, using your boot.py script from Lab 1.

Step 2: Create an Adafruit IO account and set up a test feed

Go to https://io.adafruit.com/ and create an account if you do not already have one. Sign up is on the bottom of the page. The free option is sufficient for the class. Once logged in, navigate to your IO "dashboard" and in the top center of the screen will be "IO key" in yellow. Click on this, and make a note of the "active key" displayed.

Then, set up a *feed* (a.k.a. an MQTT topic):

- Click on Feeds > View All > Actions > Create a New Feed.
- Give the feed a name of your choice, and click Create. This feed is analogous to the topics that you created in Lab 4.
- Then click on the name of your new feed as it appears in your browser.
- Click on "Feed info" on the right-hand side of the page.
- Make a note of the "MQTT by key" which is the name of the MQTT topic corresponding to your feed.

Download `mqtt_adafruit_rev1_bCourses.py` from Files > Labs > Lab 5 on bCourses, and edit the following three fields:

- adafruitUsername = "Your Adafruit.io username"
- adafruitAioKey = "Your Adafruit IO key"
- feedName = "Your feed name (format: /username/feeds/feedname) – what you copied *from MQTT by key* above"

Also, we will need to use a slightly different MQTT module from the one we used in Lab 4. Download `umqtt1.py` from the Files > Labs > Lab 5 on bCourses, and copy that file to your ESP32's flash:

```
cp umqtt1.py /flash/umqtt1.py
```

Step 3: Send test message from ESP32

Then execute `run mqtt_adafruit_rev1_bCourses.py` in `shell49` to connect to the Adafruit server and send a message on your test topic. Verify that the message appears at the bottom of the Adafruit webpage associated with your test feed (i.e. https://io.adafruit.com/USERNAME/feeds/FEEDNAME)

Step 4: Send test message to ESP32

Let's also use the ESP32 to subscribe to your test feed. This capability could be helpful for controlling an IoT device remotely, or for passing important information to the device upon which it may need to act.

- In Adafruit.io go to Dashboards > View All > Actions > Create New Dashboard.
- Give your new dashboard a name and click Create.
- Click on the name of your new dashboard in the screen that appears
- Click the blue icon with the white plus sign ("Create a new block")
- Select the "Toggle" item
- Choose your test feed
- Click "Next step"
- Give your block a title and click "Create block".

Once again, execute `run mqtt_adafruit_rev1_bCourses.py` from `shell49`, and then click repeatedly on the toggle button you have created on the Adafruit website. Any toggle events during the 60 seconds after you start `mqtt_adafruit_rev1_bCourses.py` running should appear both in the terminal window where you are running `mqtt_adafruit_rev1_bCourses.py` and also on the Adafruit.io page for your feed (i.e. https://io.adafruit.com/USERNAME/feeds/FEEDNAME).

Step 5: IFTTT

According to its creators, IFTTT "is the free way to get all your apps and devices talking to each other. Not everything on the internet plays nice, so we're on a mission to build a more connected world." IFTTT stands for "If This Then That"; basically how it works is you choose an application to be an "If This" (or cause) and then another application to be a "Then That" (or effect). IFTTT has many capabilities today, and is only getting better with services and websites constantly being added to allow for more functionality. After connecting to Adafruit IO there is very minimal programming needed so you're welcome to explore everything yourself and see what IFTTT has to offer! IFTTT is a commercial platform with limited free functionality (you can create up to three applets for free).

- Go to https://ifttt.com and create an account.
- Click on the gray profile head in the top right of the screen, then "My Services"
- Click "Get more"
- Enter "adafruit" in the search box
- Click the "Services" tab below
- Click on the Adafruit icon
- Click "Connect": this will redirect you to adafruit.com and you need to sign in here with you Adafruit.io credentials.
- Click "Authorize"
- You will be directed back to ifttt.com; there should be a message at the top of the page saying that you've connected successfully. (If not, I found that it helped to click the back button on the browser and try the previous two steps again.)

Now, create your first IFTTT applet.

- Click on “Create” at the top of the ifttt.com page
- Click “Create” next to “If This”
- Click on the Adafruit icon (white flower on black background)
- You will be asked to “Choose a trigger”. Select “Monitor a feed on Adafruit IO”.
- Select your Adafruit.io test feed from the Feed dropdown menu
- Select “equal to” as the relationship
- Enter “Hello Adafruit World” as the Value (don’t include the quotation marks here)
- Click “Create Trigger”
- Click “Then That”
- Click “E-mail” (you will have to scroll down to find this icon, which is blue)
- Click “send me an e-mail”
- Click “Connect”
- Enter your e-mail address
- Enter the PIN that is e-mailed to you and click Connect
- Edit Subject and Body as you see fit, and click “Create action”
- Click Continue, then Finish

Test your applet by running `mqtt_adafruit_rev1_bCourses.py` again and you should receive an e-mail notifying you that your test feed obtained the value “Hello Adafruit World”.

Now create a second applet, where the “If This” is a *button widget* (search for this), and the “Then That” is Adafruit > then select “Send data to Adafruit IO”. Select your Adafruit test feed and enter a distinctive message to send. Click Finish to create the applet.

Now you can send a message to your ESP32 from your phone:

- Download the IFTTT smartphone app,
- Log in with your IFTTT credentials
- Touch your e-mail address in the top right corner of the app
- Touch “Widgets”
- Select your second applet (“send data to … feed”)
- Touch the Adafruit symbol (the white flower). A larger version of the Adafruit symbol will appear.
- Run `mqtt_adafruit_rev1_bCourses.py` again using shell49
- Now immediately touch the larger Adafruit symbol that had appeared.
- Confirm that your specified message appears in your Adafruit feed *and* in the terminal where you are running `mqtt_adafruit_rev1_bCourses.py`. If you see your message, you have successfully shown how you **can potentially control an IoT device from your smartphone**!

Appendix: Huzzah ESP32 pinout

Note that the two DAC pins are GPIO pins 25 and 26 – highlighted in deep yellow on the left.

Adafruit HUZZAH32 MicroPython

GPIO	ALT	µPy		
		RESET		1
		3.3V		2
				3
		GND		4
26	DAC2	A0		5
25	DAC1	A1		6
34	ADC6	A2		7
39	ADC3	A3		8
36	ADC0	A4		9
4		A5		10
5	SCK	A16		11
18	MOSI	A17		12
19	MISO	A18		13
16		A19		14
17		A20		15
21		A21		16

µPy	ALT	GPIO
VBAT		28
EN 3.3V		27
VUSB		26
A12	LED	13
A11	BOOT	12
A10		27
A9	ADC5	33
A8		15
A7	ADC4	32
A6		14
A15	SCL	22
A14	SDA	23

Boot mode:

BOOT (A11) has a built-in pull-down
Connect to 3.3V on power-up for safe boot.

EN 3.3V: tie to GND to disable regulator

Legend:

sup	ADC	SPI
GND	DAC	I2C
BOOT	VBAT/2	LED
input only!		
VBAT/2 tied to VBAT2		