## Week 1 Introduction to Course Content and Tools

**Due Jan. 24, 2025**

Submit an assignment completion report that includes some screen shots documenting that:

1. You joined the Yellowdig community and posted an About Me post there
2. Arduino IDE was downloaded and is running on your computer
3. Packet Sender is running on your computer
4. You can access the old IoT CourseEra video from the auxiliary videos folder: watch it too!!
5. You can find Tello SDK commands from the attachments in the Resources folder.

The assignment will be worth 10 points toward your overall grade.

Caption all screen shots in the assignment report, preferably above the screenshot itself.

To make this assignment even easier to complete, a Word document template is provided with placeholders for your screens.

## Week 2 EEK PCB Configuration Tests

EEK assembly should have been verified by Wednesday's class on January 22.

A Zip file is attached containing 4 ESP32 code examples that will work if the EEK has been connected correctly.

The assignment submission report should show that the LEDs, OLED display, push button switches and MPU are functioning as expected.

This assignment is worth 20 points and is now due Friday January 31.

There should be at least 10 screen shots or photos of the EEK PCB covering  the 4 assembly tests. Where there is Serial Monitor Output appearing in the IDE, that should be shown. Each of the four assembly tests open in the IDE should be shown. Where the OLED display shows information about the sketch's execution, that can be shown in a cell phone camera picture.

Each screen or photo you include should have an identifying caption in the assignment report. I encourage you to use the format used in the Week 1 assignment template.

Even though we did not assemble a Breadboard EEK, the Assembly Chapter 4 has good information on running the verification tests identified above.

Use the GPIO numbers that were used in the Week 2 deck.

**Note added Jan. 28**

Grading expectations showing assignment completion:

Score:  
/20  
Expected For Week 2 Assignment: EEK PCB Full Operation.  
Your completed progress after working with the PCB EEK and running Assembly Tests through IDE; 2 points awarded for each of 10 screen shots documenting EEK assembly  
- LEDs Blink and Serial Monitor messages shows which one: IDE view and Serial Monitor and Assembled EEK (3 screens)  
- Serial Echo runs and OLED displays message: IDE view and Serial Monitor (2 screens)  
- Button demo works and all 6 buttons report a button press in Serial Monitor: IDE view and Serial Monitor (2 screens)  
- Gesture Tester Works, LEDs on EEK light with EEK board rotation: IDE view and Serial Monitor and EEK photo (3 screens)

**Note added Feb. 2**

Attached PDF of my own screenshots documenting completed Operational Tests including IDE views that some submissions did not include.

## Week 3 Interfacing Sensors and Other Hardware

This assignment is split into 3 parts. For two of the parts, beginning code is provided. You are expected to make modifications to the code as described below.

The assignment is worth 15 points toward your course grade.

**Due Feb. 7, 2025**

Helpful information for the assignment is available at several of the links included in the Week 3 Links of the Week. These cover:

* Traffic light simulation code in Arduino
* interfacing with DHT11 sensor for temperature and humidity
* Using OLED to display weather station like environmental data based on DHT11 readings

Some sample code to get you started is attached to this assignment.

Starting with this code, please perform the following modifications.

**Traffic Light:**

Get the vertical pole connected from the breadboard to the ESP32 and working. See attached photo.

A green circuit board with wires and a red light

AI-generated content may be incorrect.

Your assignment report should include a photo of your EEK connected to the vertical traffic light pole.

Add serial println's to the sketch to write out the current color of the traffic light: Green, Yellow, RED to the serial monitor display.

Change the GPIOs so that the horizontal traffic light version on EEK uses top LEDs -- ignore blue color instead of yellow.

**NOTE Added Feb. 5**: unless you set the Yellow LED to OFF in your logic, it will now remain dimly lit.

Change timing of light sequencing: how long each LED is lit. Say what the delays are now in your sketch. I don't want the code submitted at this time.

**DHT11 sensor:**

Mount sensor to breadboard and use pin 19 for signal. Find DHT11 demo  and use it to get data in the serial monitor. See attached photo.

A circuit board with wires and a blue light

AI-generated content may be incorrect.

Your assignment report should include a photo of your EEK connected to the DHT11 sensor.

Get the example to work on your EEK board with the DHT11 sensor connected. Verify that the signal GPIO can be changed in the code and the wiring from 19 to another available GPIO number (check the Week 3 Powerpoint for available GPIO pins)

**Weather station:**

Show photo of weather display on your OLED.

Extend the code to light an LED if humidity gets above 40%, and turn it off if below 30%. Your choice of LED. Example will be shown in class. You can't use the LED shown in class.

You can get creative here -- real high, moderately high, normal, low -- use different LEDs and different percentage values.

Up to 5 points of extra credit for creative modeling of an environmental sensor scenario: suggest temperature range changes that light another LED from humidity.

## Week 4 EEK Capstone Mini-Project

Combination of LED Blink, OLED display, EasyButton switch control, PWM and MPU6050 behaviors.

Builds on logic found in the previous mini-projects.

Partial implementation of Capstone sketch provided; you are to complete the implementation as described below.

Changes needed to partial implementation include:

1. Fade bottom green LED and toggle bottom RED LED (reverse fade and toggle button behaviors) for lower RIGHT and LEFT button presses.
2. Change OLED displayed messages according to the button logic changes.
3. Change to PWM based effects for the Roll LEDs based on MPU angle gestures: Top RED and Top Green on the EEK.
4. Revert to the ON and OFF LED effects for the Pitch LEDs: Blue and Yellow based on MPU angle gestures.
5. Test your modified version after each step is completed.

**Sketch file upload is expected for this assignment and execution will be verified.**

Having your changes work as expected in **my IDE with my EEK** will be the primary component of the assignment grade.

Your submitted sketch should work with my EEK board.

Take some screen shots like you did for the Assembly Tests after each part is completed and include captions for each picture you include.

Assignment is worth 15 points and is due at close of day Friday Feb. 14.

If there are issues or questions, please post on Yellowdig.

**Extra Credit (up to 5 points)**

Add two more Button2 buttons for TOP LEFT and TOP RIGHT switches that affect the TOP RED and TOP GREEN LEDs like the bottom buttons affect the bottom LEDs.

One should fade after a press, the other should turn the LED on for 10 seconds and then turn it off.

Note that the TOP RED and TOP GREEN LEDs are now being handled as PWM outputs via analogWrite() method calls after the modifications made in step 3.

To make an LED being handled with analogWrite fully ON, use (for example):

analogWrite(LED\_RIGHT, 255);

To turn such an LED OFF, use for example:

analogWrite(LED\_LEFT, 0);

## Week 5 WiFi Exploration

This assignment is worth 20 points toward your semester grade and is due February 21.

You are expected to run a selection of examples exploring the WiFi coverage and features available through the EEK. Most of these were shown during week 5 in various forms. The source code for the examples is in the Week 5 zip file available from the item above.

You are also expected to show that your EEK ESP32 was registered with VUPlay. The first item below says what's expected.

I am not asking for any code changes in the WiFi examples, just the personal execution of the examples using your own EEK. Make sure code inside the examples is correct: you are including the right WiFi access credentials when you are not using WiFi Manager.

To help guide your completion of the assignment, I am including a PDF file with screen captures from my steps through the assignment. My screenshots reflect working through the examples several times over the last few months so yours may differ based on recent code changes to the examples.

Assignment steps include:

1. VUPlay registration: screen of registered devices with MAC:  (text or screen with your device MAC address and snap of completed registration)
2. Hello Web Server with included home or VU credentials: IDE and Browser screens; minimum of 2 screens
3. Hello Web Server with WiFi Manager: IDE and Serial Monitor screens; minimum of 2 screens
4. SimpleTime with OLED and WiFi Manager: IDE and OLED photo; minimum of 2 screens
5. UDP Server with OLED and LEDs: IDE and Serial Monitor and PacketSender; minimum of 3 screens
6. Dual AP and Station mode example where you connect your computer WiFi to the EEK access point; minimum of 2 screens
7. SimpleTime with Dual AP WiFi (no OLED): IDE and Serial Monitor and PacketSender; minimum of 3 screens
8. Full UDP Server and OLED and LEDs with Dual AP WiFi: IDE and Serial Monitor and PacketSender and photo; minimum of 4 screens

Please post to Yellowdig any questions or problems that you encounter.

## Week 6 AWS Demo (not assigned Spring 2025)

This was the Week 6 assignment in the Spring 2024 Villanova course where AWS IoT Core was covered.

**If you want to get course credit for learning and using AWS IoT Core you could make this the core of your semester research presentation.**

*But rather than submit an assignment report, you would create a presentation and possibly record a video of you operating the software and showing effects like those that are shown in Chapter 8 of the text.*

Start with the attached assignment zip folder containing both the sketch file as well as certs.h and utils.h.

Unzip the folder and load the sketch file into the Arduino IDE.

Find the Replace\_Me strings in the files in the project, especially certs.h.

Put values into these slots that correspond to your AWS account and IoT Core thing creation values.

First make sure that you can compile the project after making the changes and you can connect to IoT core.

In your assignment report include some screen shots/photos where you show your replaced contents and that the code running on your EEK can send and receive messages to and from IoT Core.

Do not proceed until you can see results!! Use Yellowdig, or class next week, if there are questions or issues.

You are to make the following alterations to the project:

* Implement for the top buttons (buttons 3 and 4) like what is done for the bottom buttons: code mods in the event handler and needed initialization steps
* Add packet handler for UDP packets: 1button and 2button that execute the button handlers for the bottom left and right buttons, like the mpu packet causes the button 6 (middle right) button handler to run
* toggle bottom row of LEDs if the message payload contains "hello" -- case sensitive, code already handles "Hello"
* toggle Blue LED if the message payload contains "blue"
* make middle left button cause the MPU temp to be sent to IoT like the temp packet via UDP does

As extra credit options (worth up to 5 extra assignment points):

* Change the toggle LED function to be more visually appealing, maybe with a flicker effect, or a PWM effect rather than ON then OFF
* Add a new function to publish Roll, Pitch and MPU temp as a single message that is called from the button 6 handler OR the mpu message: 3 name/value pairs in the single message, 4 if you include the EEK MAC address. You can use the following code as something to start with:

void publishMPUMessage(String roll, int rollValue, String pitch, int pitchValue, String temp, int tempValue) {

     StaticJsonDocument<200> doc;

     doc["MAC"] = macAddress;

     doc[roll] = rollValue;

     doc[pitch] = pitchValue;

     doc[temp] = tempValue;

...

}

Call this function when button 6 is pressed or the mpu packet comes from PacketSender.

## Week 6 Arduino IoT Cloud HelloWorld

The Week 6 assignment is an Arduino IoT cloud Hello World assignment.

The assignment is worth 20 points and is **due before Spring break**: February 28.

A main sketch file is attached that is my realization of this assignment that you can use but you need to start with your own sketch created in the cloud with embedded device credentials and variable definitions in two related header files. This is shown in the video attached to the Week 6 materials. Basic setup will also be shown in class. You need to create an Arduino IoT cloud account and set up the Arduino cloud agent before proceding with the assignment proper. See the account creation video.

Through your cloud account you

* create a device,
* a thing,
* associate the thing to your device where you specify cloud variables that can be read and written,
* set up cloud and WiFi credentials,
* check that there was a sketch created
* through the Desktop IDE, download the sketch
* complete your sketch logic using the additional lines in the sketch attached here
* attach the DHT11 sensor to your EEK, making sure all wires are connected to the proper pins
* upload the sketch to your EEK

Once the sketch is uploaded, create a simple HelloWorld dashboard (demoed in class) and check that the dashboard reflects the state of the blueLED and temp cloud variables. Check that you can turn the blue LED ON and OFF via the dashboard or with the top right button.

Your assignment report with included screen shots or EEK photos must show that these steps have been completed, and you have corresponding correct behavior with your Hello World application.

You can get up to 5 points of extra credit by adding dashboard widgets (gauge or chart) and reporting on your mobile device rendition of the dashboard.

## Week 7 IoT Cloud Room Monitor

The assignment is to basically implement the Arduino IoT Cloud Room Monitor example as presented in Chapter 9 of the EEK book.

The assignment is worth 25 points and is due *after* Spring break: **March 14**. Submitting early is possible.

You must make a room monitor thing in the cloud that can show the status of two EEK LEDs (Blue and Yellow) and enable switch control of them. The EEK should be able to use buttons to also control the LEDs ON/OFF. You must use the DHT11 sensor to monitor Temperature and Humidity and display warning LEDs when the values are too high. These LEDs should show on the dashboard and on the EEK LEDs. Use the two RED LEDs on the EEK.

Starter code is included here that implements the Blue LED control and Humidity display and High Humidity warning. You must add Yellow LED control and Temperature display and High Temperature Warning. But you must first add your own thing to the cloud and declare the needed cloud variables.

First only implement Blue LED and Humidity behavior using my code. You need to define the dashboard for this starter example. Two pictures are attached that show the dashboard arrangement that I came up with.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a device showing a graph

AI-generated content may be incorrect.

Show that this starter example works for you. After verifying that the example shows expected behavior, push your starter example up to the cloud. Then add variable declarations to support for Temperature and Yellow LED behaviors. Then pull the sketch down to your computer and add in the logic necessary. You can then add the new dashboard widgets once the code is working.

In the end, the dashboard needs to display all relevant indicators and controls. Such a dashboard is shown in the book chapter.

Submit an assignment report showing your progress on completing the assignment with labeled screen shots and perhaps a few EEK photos.

AND submit your final .ino sketch file so I can take your code and run it on my computer using my EEK and my dashboard. Your cloud variable names must match mine for this to work:

String elapsedTime;

~~int humidity;~~

bool blueLED;

~~bool highHumidity;~~

bool yellowLED;

int temperature;

bool highTemp;

The first four of these are in the starter example and you should use these names when you define your first set of cloud variables. Add the other three after the code and initial dashboard are working also using these exact names. **BUT see note added below**.

For extra credit, one possibility is to integrate the Network Time Protocol into the sketch logic, replacing the elapsed time string with the current day and time using NTP. Some of the previous sketch examples covered this semester showed how to format a time string from the NTP structure. You could replace the logic inside of the getReadableTime function to get the current real time as a string. You need not replace the cloud variable name itself even though the real time is not elapsedTime. Up to 5 points of extra credit can be earned.

**NOTE**: Extra credit will be awarded ONLY if the assignment is turned in by the due date, but weekends after the due date do not accrue lateness deductions.

**NOTE added March 10:**

Jack McIntyre experienced a free plan limitation where you can't declare more than 5 variables for a thing.

So I am modifying the assignment to say to remove Humidity Processing (value reading and warning) and instead to do Temperature processing.

Ask on Yellowdig if there are other issues I am not aware of.

Sorry for the inconvenience.

## Week 8 No Assignment

Midterm Exam posted and completed instead.

## Week 9 Tello-Sim Usage Quiz

Use the resources attached here to help you complete the Week 9 Assignment Quiz. The "ini" file is the source of a test button panel that will be shown in class and is also shown in the text. Use this to generate your own test button panel for testing your Tello-Sim.

The Tello-Sim source file has two lines commented out that determine the serial number and SSID and will not compile without you providing values there after uncommenting. You should keep the original version of the Tello-Sim firmware on your Tello-Sim until after you complete the quiz. You can still use the Serial Monitor to observe output from the Tello-Sim with the firmware as distributed.

In the attached "png" file numbered circles are overlayed on the 10 LEDs on the Tello-Sim. Use these numbers to answer questions about LEDs that are lit when specific Tello flight commands are sent to the Tello-Sim.

A circuit board with wires connected to it

AI-generated content may be incorrect.

The commands that are sent from the test panel buttons have string values you can see inside the list of commands once the ini file is imported into Packet Sender. Refer to these when answering the quiz questions.

Observe the log file section of PacketSender to see command responses from Tello-Sim. The values seen there will be needed to answer some of the quiz questions.

|  |  |
| --- | --- |
| What is your Tello-Sim Serial Number: \_\_\_ | DEVSNTELLOSIMA0 |
| What is your Tello-Sim SSID: \_\_\_ | TELLO-SIM000 |
| Using LED numbers in the Assignment Info picture, what LED(s) are lit when the Tello command: command is sent: \_\_\_ | 0 |
| Using LED numbers in the Assignment Info picture, what LED(s) are lit when the Tello command: takeoff is sent: \_\_\_ | 9 |
| Using LED numbers in the Assignment Info picture, what LED(s) are lit when the Tello command: up 50 is sent: \_\_\_ | 1 |
| Using LED numbers in the Assignment Info picture, what LED(s) are lit when the Tello command: ccw 90 is sent: \_\_\_ | 4 |
| Using LED numbers in the Assignment Info picture, what LED(s) are lit when the Tello command: right 50 is sent: \_\_\_ | 6 |
| Using LED numbers in the Assignment Info picture, what LED(s) turn off when the Tello command: land is sent: \_\_\_ | 9 |
| Using LED numbers in the Assignment Info picture, what LED(s) are lit when the Tello command: rc 0 0 0 30 is sent: \_\_\_ | 2 |
| Using LED numbers in the Assignment Info picture, what LED(s) are lit when the Tello command: go 40 40 is sent: \_\_\_ | 5 and 8 |
| Using LED numbers in the Assignment Info picture, what LED(s) are lit when the Tello command: motoron is sent: \_\_\_ | 9 |
| What response is seen in the Packet Sender log when the command: battery? Is sent: \_\_\_ | 90\r\n |
| What response is seen in the Packet Sender log when the command: sdk? Is sent: \_\_\_ | 30 |
| What response is seen in the Packet Sender log when the command: speed? Is sent: \_\_\_ | 100\r\n |
| What response is seen in the Packet Sender log when the command: temp? Is sent: \_\_\_ | 34\r\n |
| What is your Tello-Sim Serial Number: \_\_\_ | DEVSNTELLOSIMA0 |

## Week 9 Tello Out of the Box Activity

Take a look at these two videos (links to saved videos provided):

[Auto Flight video](AssignmentAttachments/TelloFlight1080.mov)

[Tello-Sim video](AssignmentAttachments/Tello_Auto_Client_Sim_1080P.mov)

There is an EEK sketch attached to this item that will be covered in class allowing you to fly Tello and Tello-Sim.

The commands to actually takeoff, do a few flight actions, and land are commented out. Uncomment, upload to your EEK and be sure the Tello is ready to fly before proceeding. Run this sketch targeting the Tello-Sim board BEFORE trying with the Tello. Then run with your real Tello.

NOTE: you first need to learn what your Tello-Sim's SSID is. Mine and each of your Tello-Sims have slightly different SSIDs.

If you have the time to do this activity by Wednesday and report issues to Yellowdig or during Wednesday's class that would be great!

Otherwise try to complete the activity this week sometime. Submission as an assignment is not expected.

## WEEK 10 EEK MPU Controller Version 1

The basic goal is to fly the Tello and Tello-Sim with the EEK Controller using buttons and MPU gesture actions.

A Version 1 EEK Controller sketch is attached to this assignment with a couple of incomplete code segments that you must complete before preparing your assignment report.

Two additional phases of the controller are planned in future weeks, one using the MPU and one using the Gamepad, with each one using velocity-based commands rather than discrete motion commands like you see in this example.

Explore this program and find at least 10 areas to document with screen shots in your assignment report. Please include captions for the screenshots that you include to say what area the screen shot covers. You can show code segments in the IDE as well as Serial Monitor traces of flight actions being executed both in the EEK and the responses in the Tello-Sim. Showing your real Tello in flight is not required.

Features to document are:

* Takeoff and Landing
* Drone elevation changes (up/down)
* Drone rotation (yaw) changes
  + Complete CCW motion logic based on implemented CW logic
* Drone forward/backward flight motion
  + Complete the "back" logic in response to a gesture
* Drone roll right/left flight motion
* Battery Level reporting including LEDs for relative battery strength
* Emergency Landing
* Enabling the use of the MPU to allow gestures to control horizontal flight actions

The assignment is due **April 4** and is worth 10 points.

MPU support comes from the AdaFruit MPU6050 library rather than the MPU Light Library used in the Gesture Tester example. We will switch back to the Light library in a future controller version. You need to angle the EEK board frontside down quickly and then bring back to level to cause a forward motion and backside down (quickly back to level) to cause a backward motion. To cause right and left motions you need to quickly tilt the EEK to the right and bring back to level for right and quickly tilt to the left and back to level for left motions.

The built-in LED now blinks while the Tello is working on the last command sent to it and stops when the command response is received. You might want to document where this behavior is implemented.

You should test on the Tello-Sim before flying the real Tello.

## Week 11 EEK Controller with Full Gamepad Support

This assignment is **due Friday, April 11** and is worth 15 points. There is a starter EEK sketch file attached here.

Aspects of this assigment include:

* Completing code in the To-Do areas of the sketch. There are 7 of these (as of April 1).
* Testing your code after your modifications with at least the Tello-Sim but some testing with your Tello is recommended.
* Writing up a short assignment report showing the work that you did.
* Submitting the report AND your completed code.

I will test your code against my Tello-Sim and verify that all statements needed to complete each To-Do are working correctly. I will also test some of the other areas of the sketch to make sure that your modifications did not break any existing behavior.

Please post any questions or issues to YellowDig sooner rather than later.

## Python and EEK Getting Started Activity

With your EEK and your Tello or Tello-Sim, run the two flight\_plan Python scripts attached to the Week 11 Materials.

There are some relatively minor differences in behavior and configuration between Mac and PC. For EEK and Tello-Sim integration, one of the more noticeable ones is how you name and work with serial ports. We saw this in Monday's class.

After installing Python, you should use the pip Python library manager to install pyserial:

  python -m pip install pyserial

Then run:

  python -m serial.tools.list\_ports

The Arduino IDE cannot be connected to the Serial Monitor when you want to run Python scripts that use the same Serial Port used by Arduino sketches like the EEK Drone control examples. You can close the Serial Monitor tab from any open IDE window; the IDE and the Output sub-window can remain open.

The basic Python command line in a PC or Mac terminal is:

python script.py

If you get a chance, please respond to a YellowDig poll item about your experience with this activity.

## Week 12 EEK Tello Controller Advanced MPU, Flight Memory, Python Scripting

This assignment is **due Friday, April 25** (*after* Easter Break) and is worth 20 points. There is an ESP32 sketch file that implements the basic features covered in this assignment attached here.

In addition, the Python serial interface can be used to script flights using the EEK. Some sample scripts are also attached here. These were covered in recent weeks and in chapter 11 of the text.

Aspects of this assigment include:

* MPU support expanded from previous Week 10 assignment.
* Flight commands sent interactively through EEK controller are saved to internal file system storage and then can be re-sent to the Tello/Tello-Sim to fly the mission again automatically.
* Serial port interface providing scripted flying through Python serial to EEK serial connection.

The 2 main parts of the assignment include the following. I recommend completing them in this order. You should make one assignment report submission (Word or PDF preferred) that documents both parts. I am **NOT** expecting sketch or Python script file submissions so be sure to include screen or text captures to show your work and results. PLEASE, PLEASE provide captions for all screenshots.

**EEK MPU support with Flight As Previously flown memory**

Change rc speeds for all button actions to be 50 cm/s instead of the current ones (60 cm/s?) -- be sure to change the speed in all places where buttons are handled.

Change the Pitch and Roll strings for gesture flying to be between 20 and 40 instead of between 20 and 60 and use 30 as a midpoint instead of how 40 is used as a midpoint in the attached code. (safer flying this way). The constrained tilt angle range should be left as-is: between 30 and 50.

Change the delay strings in the appendFile functions in the Takeoff button pressed function from the current 2 to 4 (some are already 4).

*For these  code changes your assignment report should include code snapshots taken from the IDE as needed to show your work: where you made code changes. Also, include some Serial Monitor or OLED photos documenting that these changes work.*

Include screen shot(s) illustrating how processCommand and appendLastCommand work to allow recorded flights to be re-flown. Using Serial Monitor output images is recommended. Before, in earlier code versions, run\_command was called directly from the button handlers for example.

Include screenshots showing how the kill button handler can be used to re-play the last successfully flown flight.

Include screenshot(s) to show that a recorded flight file is NOT saved in case the Kill button is used to abort a flight in progress.

*Also include some Serial Monitor or OLED photos documenting that a successful flight can be re-flown after it has completed.*

**Python Script design and execution**

Install python and pyserial onto your regular computer (python version 3.13.2 is recommended) and run some flight action scripts. Running scripts similar to these will be demonstrated in class. The EEK code from this assignment should be running on your EEK.

One script should be to execute motoron for 5 seconds and then motoroff to top the motors. *Show terminal output where the script was run*.

One should include safe, non-rc commands like we used for the run\_flight\_plan function in our early flight testing: takeoff, fly a little, land. *Show terminal output where the script was run*.

One should be for a safe flight where the Tello flies a counter-clockwise 360 degree circular arc using an rc command -- be sure to have enough floor space to safely execute such a flight. Then change the command to fly the circle route in the clockwise direction. In each case, point your Tello's nose in the direction you want to fly. You should need to change the Yaw parameter only.  *Show terminal output where these scripts were run*.

Note: scripts should have short delays when rc commands are being used and sending an "rc 0 0 0 0" command will cause the Tello to hover after the last command was executed.

Please post issues and questions on Yellowdig.

If clarifications are needed, I will post some notes here at the bottom of this assignment description.

You can primarily test your work against the Tello-Sim device rather than a real Tello.

## Week 13 EEK and DroneBlocks Simulator

This assignment is due **May 2, 2025.**

Chapter 12 of the text presents the DroneBlocks Drone Flight Simulator based on the Tello drone SDK. Be sure to read this section of Chapter 12.

The simulator is implemented as a web page, and you have to give your browser permission to send commands to the simulator. The GitHub page link below explains this. The use of the Chrome browser is assumed.

There is a Python library that needs to be installed for your local Python. This will be shown in class. You should use the Python Virtual Environment (venv) feature as will be shown in class. This is described in the text as well.

Helpful links for this assignment include the following:

[DroneBlocks Tello Simulator](https://pypi.org/project/DroneBlocksTelloSimulator/)

[DroneBlocks MQTT Simulator](https://coding-sim.droneblocks.io/)

[GitHub DroneBlocks Simulator](https://github.com/dbaldwin/DroneBlocks-Tello-Simulator-With-Python)

Be sure to review the content contained on these pages before attempting to complete the assignment. Using a Python venv is described and recommended.

The assignment requires that you set up your Python environment and the Chrome browser configuration for so-called insecure content.

To show that the installation steps are complete, execute the Python script simexample.py. You will need your simulator key entered into the source file. We will also show this in class.

Experiment by adding some Tello commands to the command list in the Python file. Show your results in you assignment report. The drone leaves a yellow-line flight trace as it executes commands it receives. You can reset the simulator view after each test run. *Do not start the rest of this assignment* until you can fly the DroneBlocks simulator with Python alone.

Once you can fly the simulator with Python alone, a combination of a new EEK Arduino controller script and a new Python bridge script (both are attached to this assignment) allow us to use the EEK to fly the simulated drone as well. We can attach flight plans to Gamepad controls and then fly those flight plans by pressing the control (like Joystick or Shoulder button).

However, there is a run-time error that is raised when running the flight plan causing the EEK to restart. Fortunately, the EEK recovers and we can keep communicating wth the Python bridge without the need to restart it. You should document this restart in your assignment report.

Since this is the last assignment of the semester, you can explore this kind of EEK control on your own and decide what to document. I do not expect your code to be submitted -- just the assignment report. You might change the flight commands in one or more of the flight plan functions. You must show some manual EEK Gamepad flying as well (we do NOT use rc commands) and at least one gamepad control flight plan.

**Extra Credit**: Up to 5 points. Modify the Simulator library in your venv  directory to allow the simexample.py example to "fly" my Tello-Sim rather than the DroneBlocks one. Show the Tello-Sim's Serial Monitor output when getting commands from the Python. The file that you need to modify will be shown in class.

**Note added April 27**

The Controller Python and Arduino files were modified based on some testing in a Windows environment. Check out the thread on Yellowdig about EEK and Python communication arrangements.

**Note added April 28**

Small modification to the EEK to Python Arduino sketch file so that the source UDP port is not set when sending messages to port 8889. This may help PC users with connectivity.

**Note added April 30**

See Week 15 class materials deck for useful info for PC users about this assignment regarding firewall permissions.