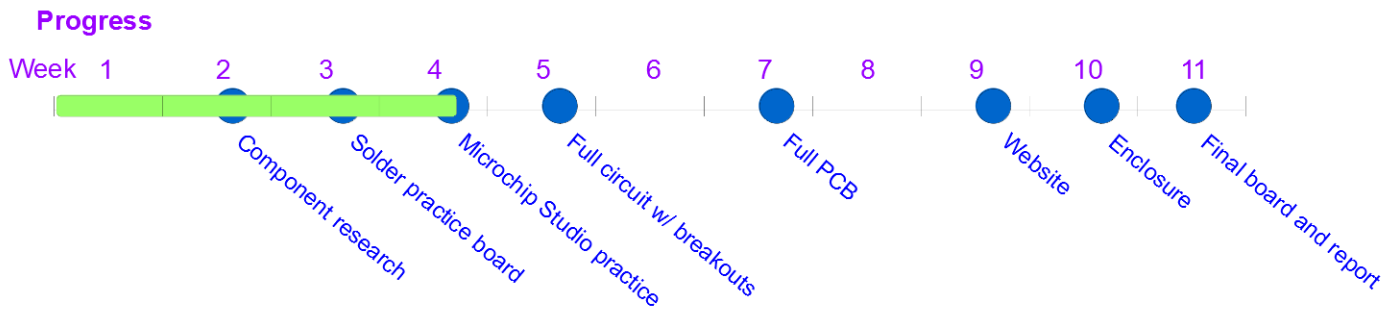
EE 326: Electronic System Design 1

Task 4: Full Webcam using Breakouts

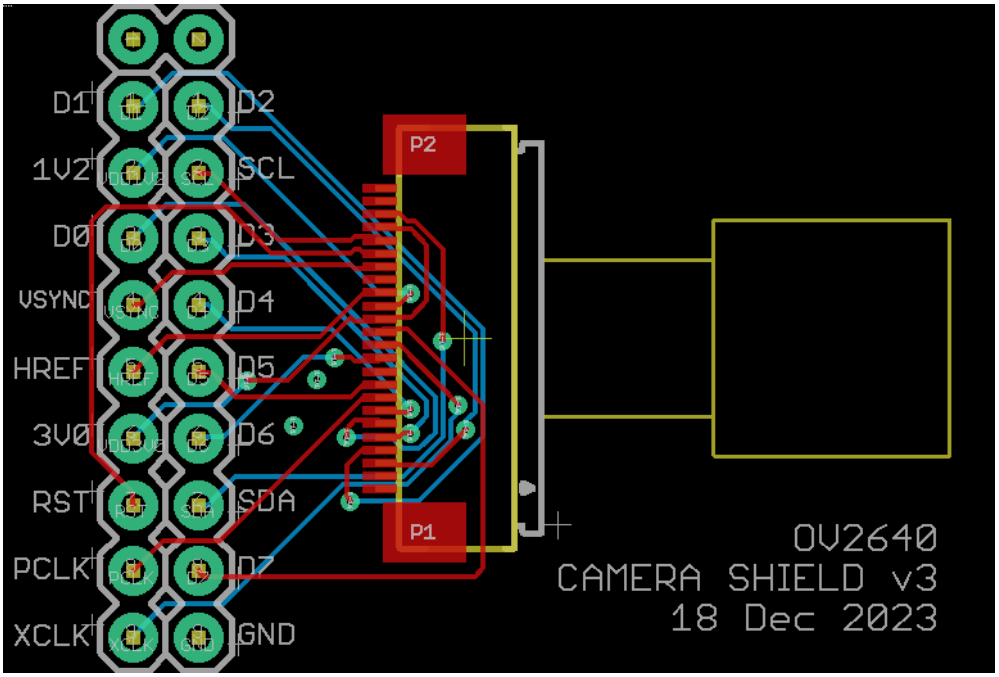
Due: Feb 6, 2025



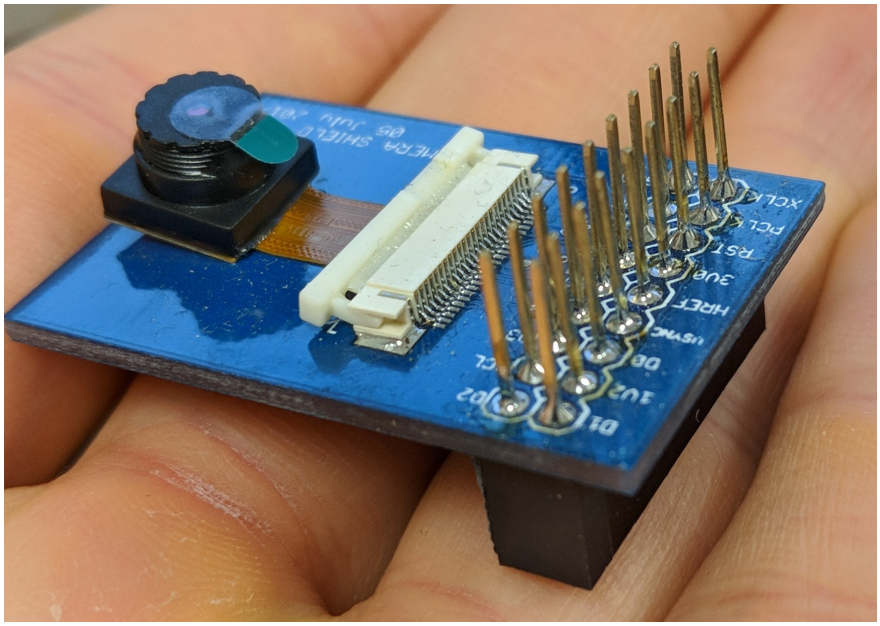
Please complete this assignment with your group. To get credit for your work, please record a video of your operational circuit. Please make sure to include at least the following.
1) Your full circuit, pointing out and narrating the connections that are asked for in the assignment.
2) Your screen showing the tests being passed in real time (i.e. press reset on your MCU and show the screen as it performs the tests).

For this assignment, you will create your entire webcam system using breakout boards and through-hole components.

1. Construct the breakout board for the OV2640 (camera).
   (a) Use the layout shown below for component placement.
   - Make sure to solder the big pads of the camera connector.
   - I would recommend to use the soldering iron for the camera connector, as opposed to the hot air gun, because you will very likely melt the plastic.



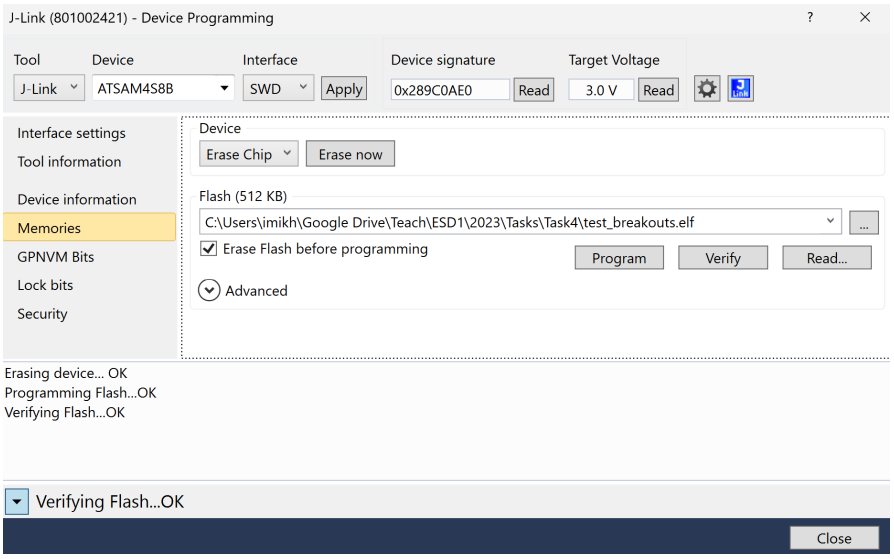
(b) Use tall female headers for the camera board.
   - Make sure that the female parts of the pins are facing down! This is shown in the image below.



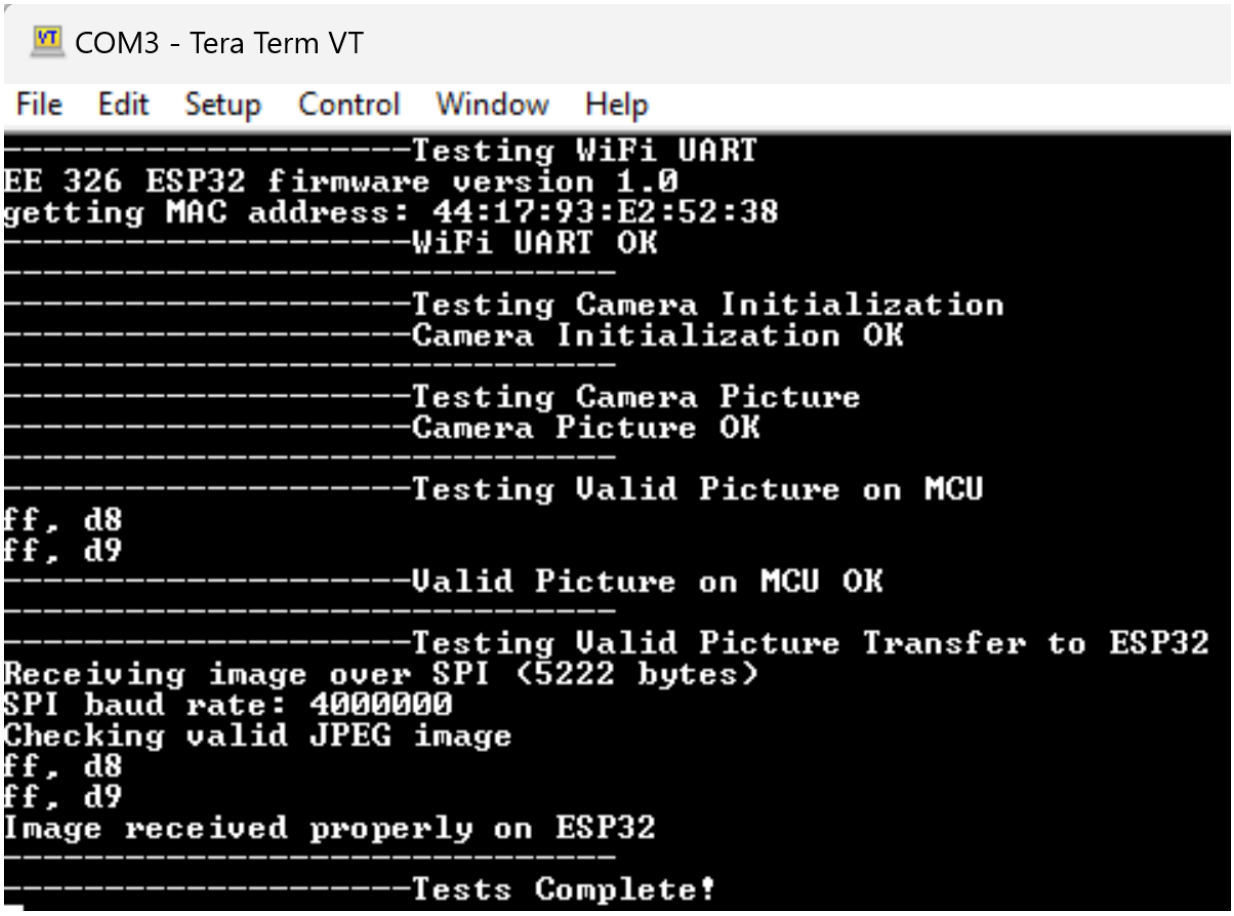
2. Create the webcam using the breakout boards that you have created, as well as with breadboards, jumper wires, and through-hole components (that you can put in the breadboard, such as resistors, LEDs, and buttons). Make sure you do at least the following:
   - Connect power and ground to all components. The power source is the USB-to-UART module, which gets regulated to 3V with the regulator on the ESP32 board. Then, you can use the 3V and GND pins to power everything else. Remember also that the MCU has a built-in voltage regulator, so make sure to connect the jumper in order to use it to power the core and PLL.
   - Connect GPIOs 15 (RX) and 13 (TX) of the WiFi to USART0 of the MCU. The native baud rate of this data bus is 115,200.
   - Connect GPIOs 25, 26, and 27 of the WiFi to the anodes of the onboard LEDs. These will be indicators.
   - Connect GPIOs 21, 22, 23, and 32 of the WiFi to free GPIOs of the MCU. These will be for control signals.
   - Connect the RESET pin of the WiFi to a free GPIO of the MCU. The MCU will be able to reset the WiFi.
   - Connect GPIOs 19 (SCK), 17 (MISO), 18 (MOSI), and 16 (CS) of the WiFi to the appropriate SPI pins on the MCU.
   - Include a pushbutton to reset the MCU asynchronously and connect it appropriately. When it is pressed, the MCU should reset.
   - Connect a pushbutton to a free GPIO of the MCU, and the other end to ground. This will be used to put the WiFi chip into setup mode.

   Note that some of the MCU peripherals have more than one pin they can run from. For example, as seen on page 52 of the datasheet, PCK1 is available on PA17 and PA21. For a given pin, you can use any one of its Peripheral functions, but only one per pin. For example, PA17 cannot be both PCK1 and PWMH3. You should first connect the components that have no choice, then those that have some choice, and finally those that have numerous choices. In order to figure out which pins on the MCU are available, refer to the layout of the MCU breakout in Task 3. Any pins that are connected to the camera are not available.

3. Test your work by loading the WiFi firmware onto the ESP32 and running some tests on the MCU. The steps are detailed below.
   (a) First, you will load the WiFi firmware. To do so, you have 4 options, detailed below. In any case, the code consists of two parts: the firmware and the file system. The firmware controls the operation of the chip, and the file system contains the files that are hosted on the webcam website. You will need to load both to get proper operation of the final webcam, but for this Task you really only need the firmware.
      i. For the first option, you can upload the code and file system as binary files using an OTA (over-the-air) updater. To do this, first install Arduino (preferably version 2 and up). Then, install the ElegantOTA library through the library manager. Next, load the "ee326_esp32_load_firmware.ino" code, available on Canvas, and modify the WiFi parameters in the code as appropriate. Finally, load the code onto your ESP32 through USB. You can find more details of this process here.
      Once the code is loaded, open the Serial Monitor in Arduino, and you should see an IP address. Go to "YOUR_IP_ADDRESS"/update, and you should see a screen that will allow you to upload new files. First, upload the file system using the "ee326_esp32_filesystem_v1.bin" file provided on Canvas. Next, upload the firmware using the "ee326_esp32_firmware_v1.bin" file provided on Canvas. That's it!
      ii. For the second option, you can upload the code and file system as binary files using an OTA updater, but from the command line. First download and install Python. Then, download the file espota.py. Next, load the "ee326_esp32_load_firmware_commandline.ino" code, available on Canvas, and modify the WiFi parameters in the code as appropriate. Finally, load the code onto your ESP32 through USB.
      Once the code is loaded, open the Serial Monitor in Arduino, and you should see an IP address. Open a terminal on your operating system, and type
      ```
      python3 espota.py -s -r -f PATH_TO_FILESYSTEM_BINARY
      -i IP_ADDRESS -a esd1
      ```
      to load the file system. After this transfers, type
      ```
      python3 espota.py -r -f PATH_TO_FIRMWARE_BINARY -i IP_ADDRESS
      -a esd1
      ```
      to load the firmware. After the transfer is complete, you are done!
      iii. For the third option, you can set up VS Code and PlatformIO and upload the source code yourself. If you are interested in this option, here is a guide on how to get up and running. Once you have VS Code and PlatformIO working, you can load the project from Canvas and load the code onto the ESP32. Don't forget to also load the file system!
      This is the better option if you want to tinker with the code or just get a better understanding of what is happening under the hood. This will also be the only way to make custom pin assignments for peripherals such as UART and SPI. Finally, VS Code + PlatformIO is a great, professional development environment that is very extensible and a good way to get started on your own projects with embedded systems.
      iv. For the fourth option, come see Ilya during office hours and he can load the firmware onto your chip.
   (b) Next, you will test everything together using the MCU. To test your work, open Microchip Studio and open and configure the Device Programming interface just as you did before. Then, go to Memories in the navigation on the left. Under "Flash", select the "test_breakouts.elf" file provided with this Task on Canvas. An example of this screen is shown below.



Before programming your MCU, open your serial terminal application and monitor the output. It is a good idea to disconnect the RESET pin of the WiFi module from your MCU for these tests. A series of tests will be run, and the results will display on the screen. Click Program in the Device Programming interface to load the program onto your MCU. If everything is successful, you should get an output like below. *Note: you will probably see a lot more on the screen than just these tests, depending on when in the boot cycle of the ESP32 you start monitoring*. You can run the test again by resetting your MCU, either using the button you have connected, or the Device Programming interface.



If you don't get anything, check your WiFi connections. If the test gets stuck after "Testing Camera Initialization", check your I2C and XCLK connections, as well as the resistors on the MCU board under the camera header. If the test gets stuck after "Testing Camera Picture", check your camera VSYNC and HREF signals. If the test returns "INVALID Picture! Check Camera Data and Control Lines" after "Testing Valid Picture", check your camera data (D0-D7) and control signals (VSYNC, HREF, PCLK).
Do not hesitate to use your logic analyzers. This is why we used the tall pins for the camera, so that it would be easy to connect the logic analyzer leads to the various signals.

FAQ

Q. I am trying to load the WiFi code using VS Code, but it is not able to connect to my board. What is going on?

A. The most common reason is that the board is not in download mode. Try pressing the IO0 button, then RESET (while still holding IO0), and then release RESET.

Q. The .elf file will not load. What should I do?

A. Try using the .hex file instead (also provided on Canvas).

Q. The OTA portal to load the ESP32 firmware does not seem to open at the prescribed IP address. What is going on?

A. This method has proven to be finicky on eduroam. It sometimes works and sometimes doesn't. If it is not working, try a different network. Otherwise, try one of the other 3 methods.