ME 461 Homework #2

100 Points, Extra Credit 20 Additional Points

Learning EagleCAD and Connecting Wi-Fi between LabVIEW and Robot

Check Off #1: Finish Tasks 1 & 2 and Optional Extra Credit (Task 4), Due by Wednesday September 27th by 1:00 PM

Check Off #2: Finish Tasks 3-8, Due Wednesday October 4th by 1:00 PM

Check Off #3: Finish Task 10, Wi-Fi Communication with Robot Car, Due Wednesday October 11th by 1:00 PM

Submission at Box, Task 9, due October 11th, by 11:59 PM

Introduction

Make sure to read through this entire document before getting started so you know all the parts of this homework and can plan accordingly.

The first goal of this homework is to teach you how to use the printed circuit board CAD software EagleCAD. I have created quite a number of videos instructing the use of EagleCAD 7.7. You can find all these videos, in addition to the install executable and other needed files, at the class lecture's Box folder under the folder "ME461ClassFiles/EagleCAD." You will be performing all the steps to build your own circuit board short of actually sending your credit card to the fabrication company "Advanced Circuits" www.4pcb.com. NOTE: You will NOT be purchasing your circuit board. I have already fabricated the base circuit board if you are interested in seeing it.

If you search online, you will find that EagleCAD is now at version 9.6 or higher and owned by Autodesk. Autodesk is working on moving EagleCAD into their Fusion 360 product. I am still using version 7.7, *one of the last versions by Cadsoft who was bought by Autodesk*, because the College of Engineering has not purchased the latest license. So, I recommend for this class that you perform this homework with version 7.7 and then when you get used to EagleCAD you will be able switch to the latest version if you need to outside of this class.

You will first watch the videos I created on how to use EagleCAD 7.7 and any other training you can find online. Note that most of these videos were recorded spring 2020 when, my other class, SE423 was

asked to stay at home due to the COVID-19 pandemic. That semester the goal was to instruct the students how to build the circuit board for their own Segbot robot based on a MSP430 microcontroller, which is different from the TMS320F28027 processor I am asking you to use for this homework. So, the circuits that I build in these videos are somewhat similar to what you will be constructing but definitely not the same. As you watch these videos, feel free to perform the same steps that I do but note that your circuit is different.

After you have finished watching my training videos, you are ready to copy the board schematic given below. This given schematic is the base schematic that each of you need to construct. **Extra Credit:** In addition to this base circuit, you will pick three sensors to add to your board. The sensors you choose are up to you but at least one of the sensors must be a surface mount device. I give some websites below that have a large number of sensors. For these three sensors, you will need to create in EagleCAD a "symbol", "package" and "device" for each. Then, use these newly created devices in your circuit board and wire them to the TMS320F28027 processor. NO, you cannot find a library online and use it. You must create your own, symbol, package, and device for all three sensors. **End Extra Credit**

Tasks

Your job is to pace yourself so you get everything done by the multiple due dates.

1. View all the EagleCAD instructional videos I recorded. As you are viewing the videos you will want to build some of the circuits that I build while recording the video to help you learn the different steps of circuit board construction. The circuit that I build in these videos is not the circuit you will be building for this homework. You do not need to turn in these "scratch" files that you create while learning EagleCAD. While learning with these videos, do not worry if you cannot find the exact same part that I use in the video. Picking a different part works just fine while you are first learning. I recommend you finish this task **by Wednesday September 20th** (note: not a check off date, just a recommendation).

2. Create all the base schematics given below. Make sure to look at the full schematic and the descriptions of the zoomed in parts of the circuit below. Here you will need to make sure you find every part in the below schematic. This is your first check off and this task needs to be checked off by your TA or instructor by **Wednesday September 27th at 1:00 PM**. This is **pass/fail and worth 10% of the homework grade**. Either you created the entire schematic, or you did not. Make sure to check off early so that you can add parts that you forgot before the deadline. **Extra Credit:** In addition, you need to have chosen your three sensors to add to the base schematic by this check off. See task 4 below. **End Extra Credit**

3. When finished with the base circuit's schematic, it is probably a good idea to work on the board layout to organize the parts generally in correct positions before adding your three chosen

sensors. Your circuit board can be as large as you need, but the idea of the base board is to have it connect to a Raspberry Pi board. Also **very important,** make sure your 20X2 pin female header on your board is orientated such that it connects correctly to the 20X2 male header of a Raspberry Pi. The female header on your board will be soldered on the bottom of the board.

4. **Extra Credit:** With the board somewhat laid out, you are ready to create and add the three sensors you chose to add to your circuit board and interface with the TMS320F28027 processor. **I need to approve the sensors you pick, so send me an email with the sensors you would like to use. This is also due by the first check off by September 27th.** I want to make sure you are not picking a sensor that is too easy or hard to attach. Then, as described in the training videos, you will create an EagleCAD symbol, package, and device for all three of these sensors. If it is a "through hole" device, you will need to indicate the outline of the sensor in your package. You should save these parts in your own EagleCAD part library so you can submit it along with your other files. Then add these three sensors to your board and wire them appropriately to your TMS320F28027 processor. Some of the wires connecting your sensor may also be connected to the 20X2 raspberry pi connector. For example, if your sensor uses the I2C serial port to communicate, you will have the SDA and SCL pins connected to your sensor and to the 20X2 connector. **End Extra Credit**

5. In your schematic, run the ERC error check. Take a **screenshot** showing that you have no critical errors. There will be some errors that do not need correcting and are just possible warnings.

6. Now that all your parts are on your board, you need to decide where to place all your parts on the board remembering that they cannot be too close together because wires still need to be routed to connect devices. You will be using 8 MIL (thousandth of an inch) width traces for most signals. For GND, +3.3V and +5V you will be using 20 MIL width traces (except where the pad connected, has a smaller width). Use the "autorouter" to route most traces on your board. For those GND and power pads that are smaller than 20 MILs you will need to hand route first and then run the autorouter. See the training videos again on how to do that. Take a **screenshot** showing your route achieved 100% routing. In addition, when you have a 100% routing, add a GND area fill to act as a ground plane on both the top and bottom layers. Take a **screenshot** of the completed board.

7. Run the DRC Error check. Watch the videos and see EagleCAD Tips.pdf to find all the settings for the Design Rules Check. Take a **screenshot** of the DRC results. You will find that there are some errors that are OK. For example, a GND wire's width is small than 20MILs at the surface mount pads that are smaller than 20MILs.

8. Go through the steps to fabricate your circuit board, but DO NOT order your board.

a. Generate your Gerber files with the given .cam Job file given at Box.

b. Zip all necessary Gerber files and send to www.4pcb.com "Free PCB File Check". The videos help with this along with EagleCAD Tips.pdf.

c. Wait for an email back from 4pcb.com say that you have "No Show Stoppers." Take a **screenshot** of that email indicating no show stoppers.

The second check off is due by October 4th at 1pm. Demonstrate that you have finished tasks 3-8.

9. Submit to your Box folder:

    a. All above requested screenshots.

    b. Gerber files zip file

    c. Zip of all your homework files including your sensors library file.

    d. PDF print out or clear screenshot of your final schematic.

    e. Clear screenshot of your final board layout and routing.

    f. Clear screenshots of the part's library Device view that you created for the three sensor's library Device view.

10. LabVIEW Wi-Fi communication with Robot Car. Task #10 is at the end of this document. It is a check off and then you need to submit to Box your LabVIEW vi file and your main *.c file for the F28379D processor. Please jump to the end of this document for task #10 instructions.

Reference Information

The following sections contain important information you may have to refer to multiple times in order to complete tasks 1-9.

Parts Needed for Base Circuit

ME461EagleLibrary.lbr (given in Box folder)
- TH_PUSHBUTTON
- FT232RL
- MINI-USB-SHEILD-UX60A-MB-5ST
- TLV1117-3V3
- TMS320F28027DAS

rcl.lbr
- C-US
  Find a 1206 size capacitor. Change its value to specify Farads in your schematic
- CPOL-US
  CPOL-USE3.5-8 100uF larger through hole capacitor. Change its value to specify Farads in your schematic. Need only one. For regulator's output.

R-US_
Find a 1206 size resistor. Change its value to specify Ohms in your schematic.
L-US
L-USL2012C

con-lstb.lbr

MA03-01	Make sure Pin 1 is a square pad. You may have to modify the associated package.
MA03-02	Make sure Pin 1 is a square pad. You may have to modify the associated package.
MA07-02	Make sure Pin 1 is a square pad. You may have to modify the associated package.
MA20-02	Make sure Pin 1 is a square pad. You may have to modify the associated package.

Pinhead
PINHD-1X2
PINHD-1X2 Make sure Pin 1 is square and Pin 2 is octagon. You may have to change the associated package.

Led
LED
LEDCHIPLED_1206

Choosing Your Three Sensors

Extra Credit

Find and choose three sensors to add to the base circuit. You will create an EagleCAD library for the sensors and wire the sensors to the TMS320F28027 processor on your board.

The sensors you choose need to be board mounted sensors in either a surface mount package or through-hole package (this includes a sensor that has already been solder to its own circuit board and then can be connected with header pins.) One of the three sensors MUST be in a surface mount package. No cables connections, at least for the three sensors you need to choose. If you want to add a fourth sensor that has a cable connection, go for it. The three sensors need to be approved by me. In your EagleCAD package for the part, make sure to add an outline of sensor in relation to its pins or pads. Even if the sensor just connects using header pins, you need to go through all the steps of creating the sensor's EagleCAD "package", "symbol" and "device."

Possible sensor websites are listed below. You can choose a sensor from a different website as well.

DigiKey sensors: https://www.digikey.com/en/resources/sensors/index?WT.z_vanity=SensorSelector

Analog devices sensor: https://www.analog.com/en/products/sensors-mems.html

Pololu sensors: https://www.pololu.com/category/7/sensors

Adafruit sensors: https://www.adafruit.com/category/35

Sparkfun sensors: https://www.sparkfun.com/categories/23

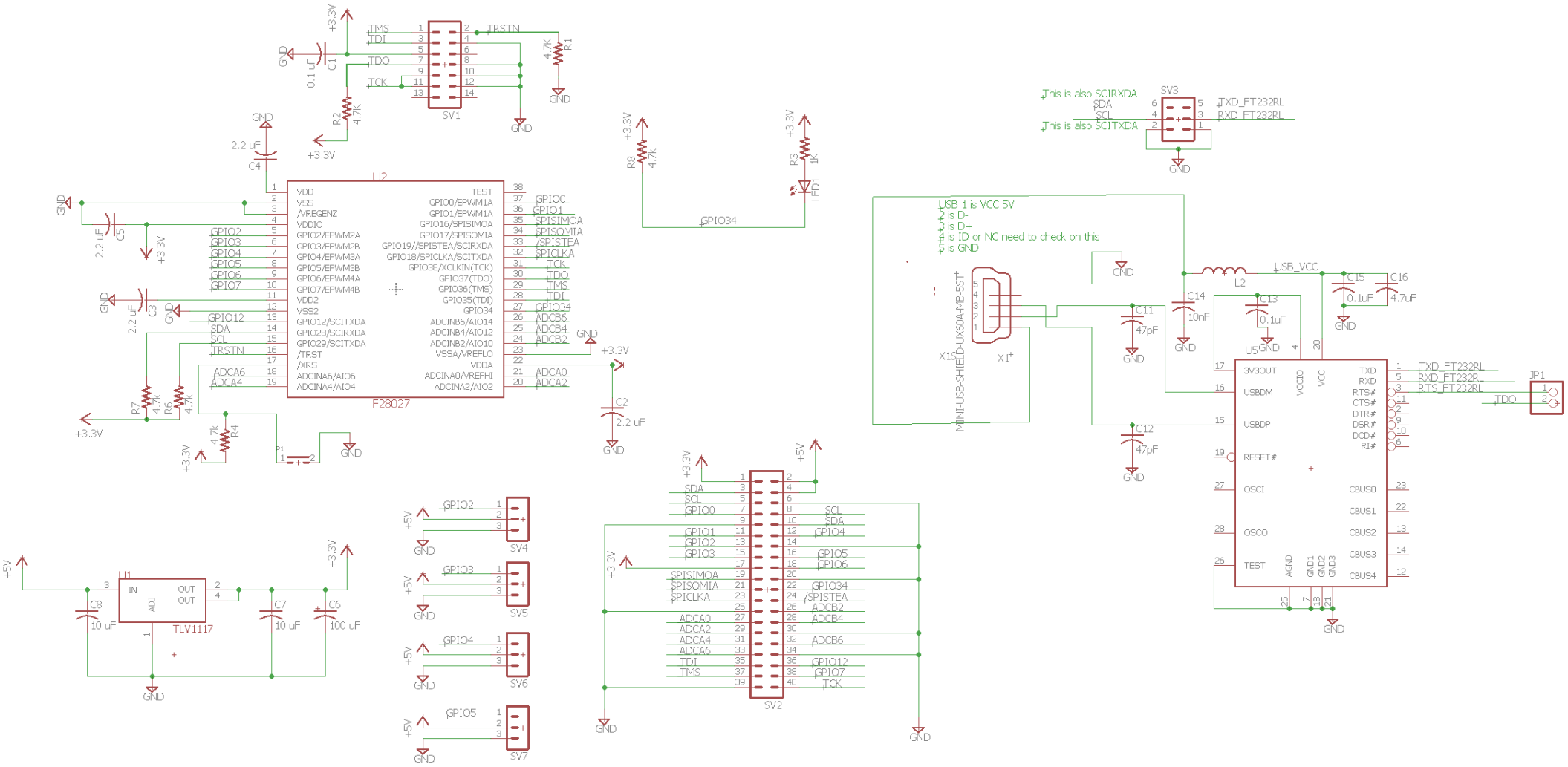
Robotshop sensors: https://www.robotshop.com/en/sensors.html

Solarbotics sensors: https://solarbotics.com/product-category/parts/sensors/

End Extra Credit

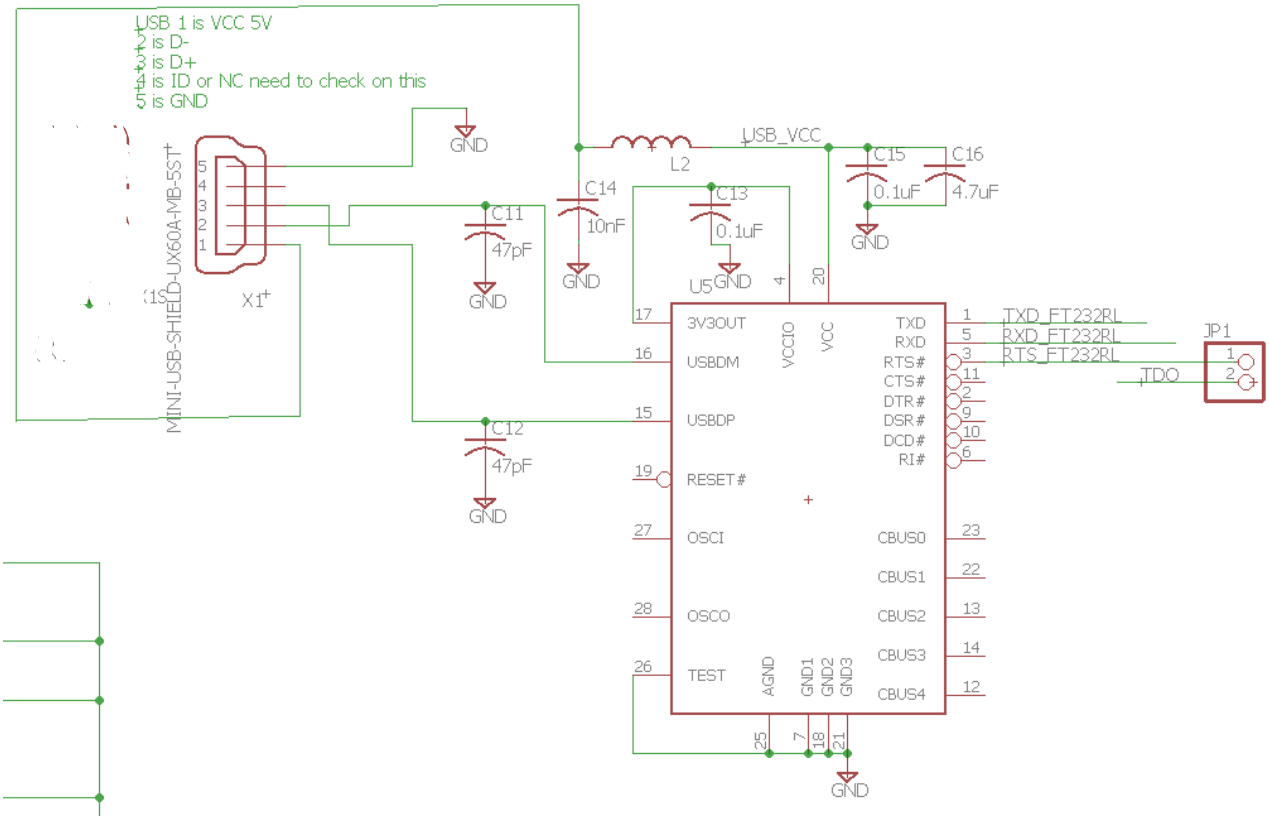
Base Schematic

You need to reproduce the following schematic. Zoom in if you cannot see some text.

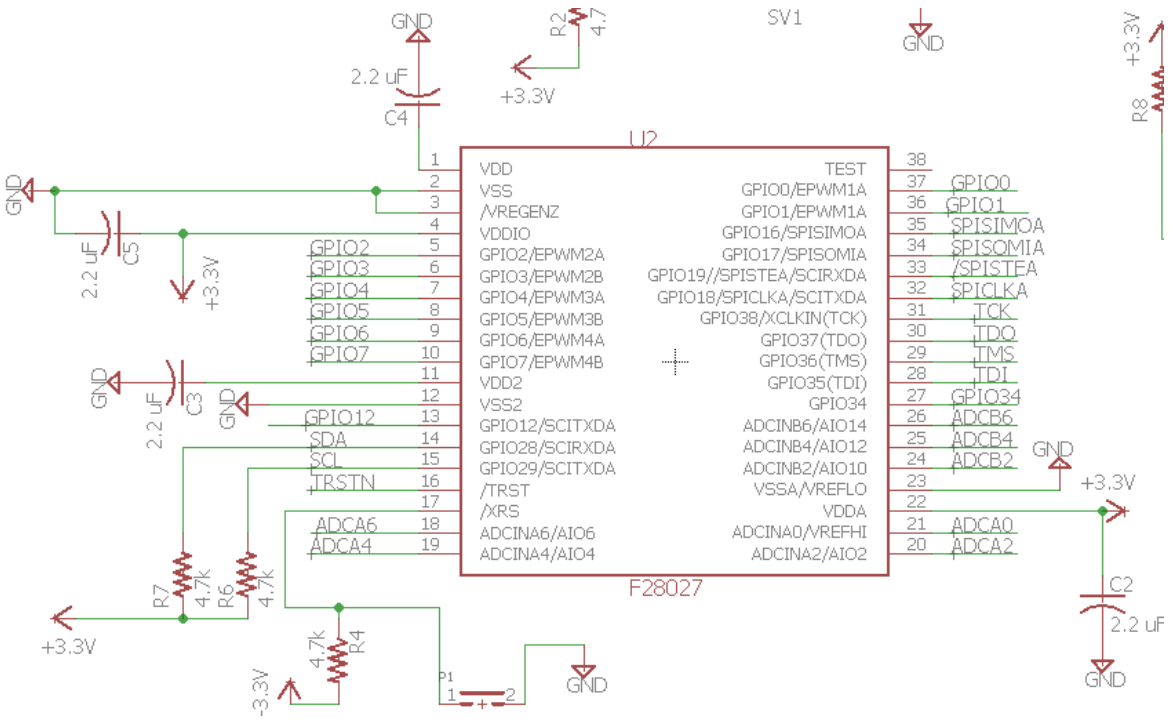


Breakout Diagrams of Parts from the Base Schematic

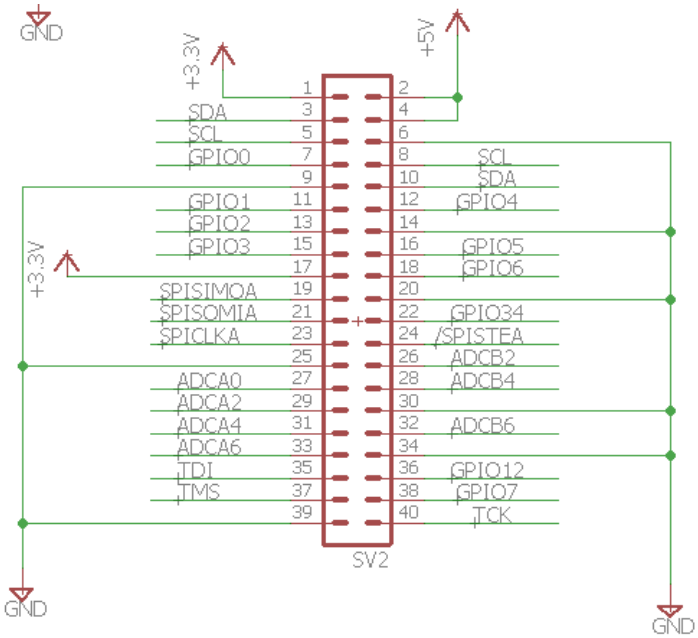
This is the virtual USB COM port circuit that converts the TMS320F28027's UART (SCIA) to USB. I recommend you start with this circuit first. The USB connector and FTDI FT232RL chip are in the ME461Eagle_Library. The rest of the parts are found in the pinhead and rcl libraries, see above. Do not worry if your chips are labeled differently. For example, the FT232RL chip is U5, but your board may have it U1. JP1 is a jumper that can connect the RTS (ready to send) pin of the serial port to a boot pin of the TMS320F28027 processor.

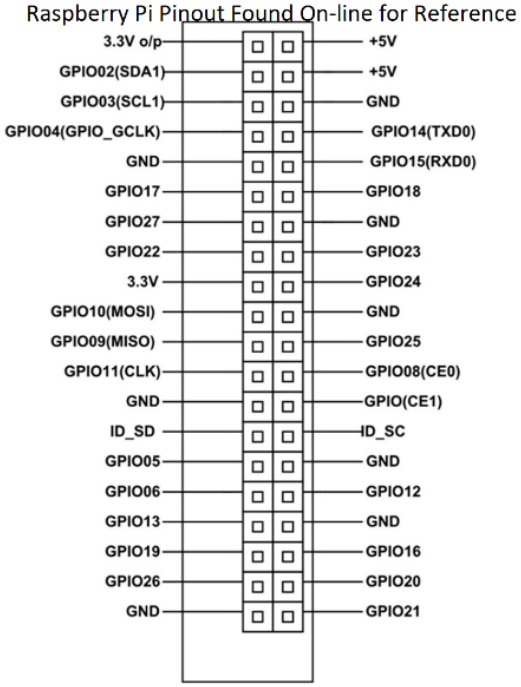


The TMS320F28027 processor. This processor is the lowest speed and lowest memory version of the C2800 processors. It also comes in a package that is "pretty" straightforward to solder and that is the reason I chose it for this board design. When you are laying out this board make sure to keep the four 2.2uF capacitors within at least one centimeter (or so) of the power pin it is connected to. This processor has an internal crystal of 10MHz and that is clocked up by its PLL to 60MHz. It is a slower processor with fewer peripherals and pins but is programmed in the same fashion as the TMS320F28379D processor.

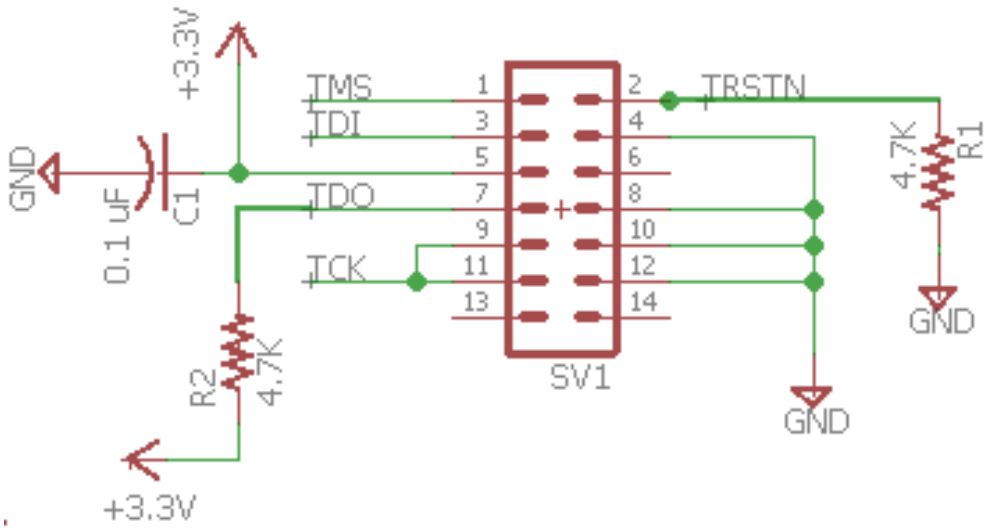


Use a 20X2 header from con-lstb library. This will be a female header soldered on the bottom of your board so it needs to be lined up with the male 20X2 header of a Raspberry Pi. Make sure you change pin one to be a square pad.

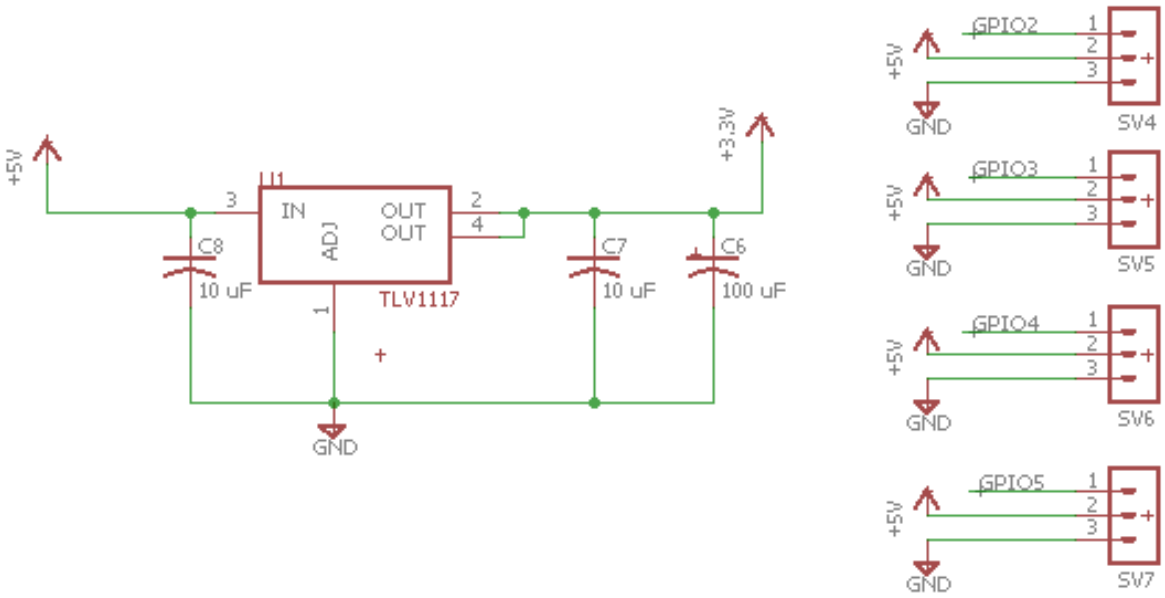




JTAG connector.

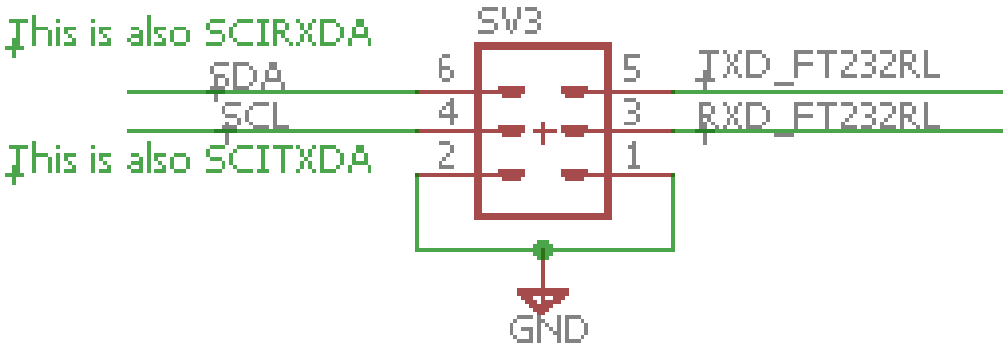


Regulator for 5 volts to 3.3 volts. The 5V would come from the 20X2 connector. This is needed if the board will be a standalone board. Also, connectors for 4 RC servos that are wired to TMS320F28027 EPWM pins.

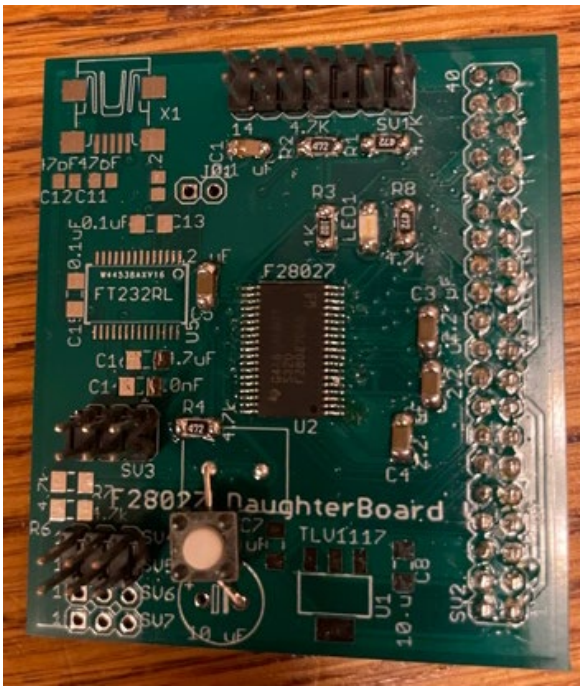


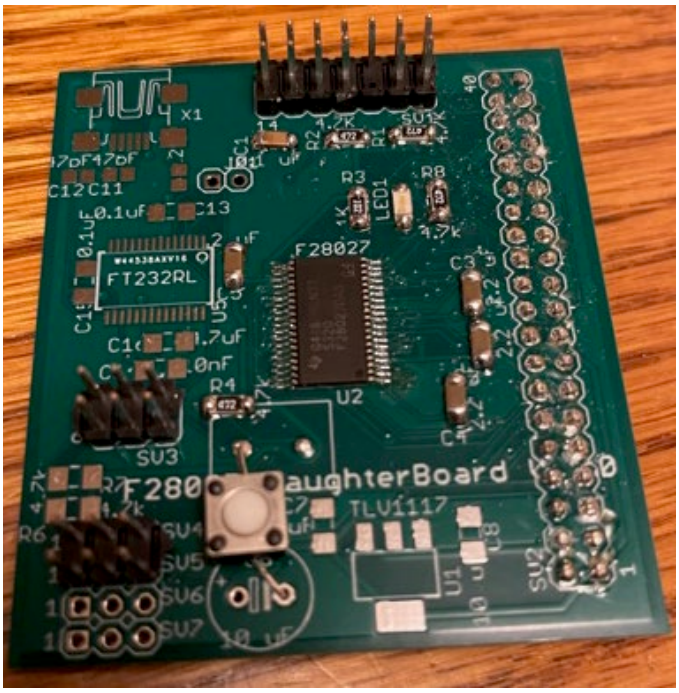
The board's one LED connected to GPIO34, which is a standard choice for C2000 development boards. GPIO34 also determines boot mode on power up. We always want it high on power up so that it why it is pulled up through a 4.7K ohm resistor to 3.3V. The 1K ohm resistor is current limiting the current through the LED to protect both the LED and the GPIO pin.

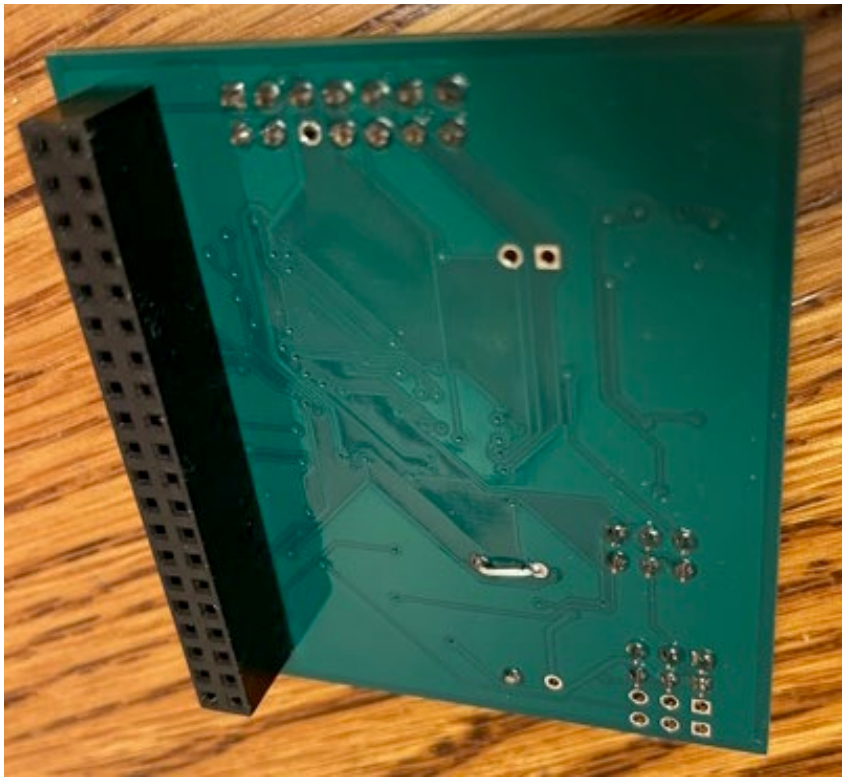
Header to either connect the onboard FT2323RL chip to the UART (SCIA) of the TMS320F28027 processor or to allow for an external FTDI board/cable to connect to the TMS320F28027's UART.



These are pictures of the base board that I fabricated. Your board does not have to be this exact size. You can make your board different dimensions and position the Raspberry Pi connector and your sensors however it works for your design.







Task #10: Robot Wi-Fi Communication through LabVIEW

This task will be performed on a lab bench robot. Remember to power your robot with the programmable power supply with 12V, 3A.

Most of the work for this final task is to create the LabVIEW program pictured below. This LabVIEW program is an extension to the programs you created in LabVIEW assignment #2. If you have not completed that assignment, I recommend you finish that assignment first.

In LabVIEW assignment #2, you were shown how to draw lines/squares in a Picture Box. In the second exercise you were shown how to communicate using the TCPIP protocol over Ethernet. The TCPIP communication talked to an Echo Server that simply sent back whatever text was sent to it.

For this task you are going to take the experience you gained from LabVIEW assignment #2 and create (copy pictures below) a LabVIEW program that receives eight floating point numbers from your robot's red (F28379D) board at whatever periodic rate your program chooses (the given code below sends these floats to LabVIEW every 50ms). These eight numbers are displayed in text boxes in your LabVIEW application. In addition, the first two numbers sent are assumed to be the X and Y coordinates of your robot car. The below given C code varies these X and Y points as if the robot was traveling in a circle. In Lab 6 and Lab 7, you will use this program to display in the LabVIEW Picture Box the actual X and Y location of your robot.

In order to have the robot send TCPIP Ethernet data wirelessly to LabVIEW, there is an addition microcontroller on the robot called the ESP32 that has Wi-Fi built in. The TMS320F28379D processor does not have Wi-Fi capability so the ESP32 needs to be added for this communication. You will see in the steps below that you will need to "telnet" into the ESP32 processor which is running an operating system called Nuttx. Easiest way to explain Nuttx is that it is a miniature Linux that can run on slower and small memory capacity microcontrollers. Once you telnet into Nuttx, you will run a C program that I wrote called "tcpLVCOM". This program waits for data either coming wirelessly from LabVIEW or coming serially through a UART serial port connected to the F28379D's UARTD (SCID). Summarizing the communication:

- If you click "Send" in your LabVIEW program, it sends eight floating point numbers over Ethernet/Wi-Fi to the tcpLVCOM application running in Nuttx. When tcpLVCOM receives the eight floats, it then sends the numbers to the red board over UART.
- Also, the given test code below sends eight floating point numbers through UARTD to the tcpLVCOM application every 50ms. When tcpLVCOM receives this new data over UART, it sends this data to LabVIEW over the same TCPIP/Wi-Fi port. Then these eight numbers are displayed in text boxes and the first two numbers are also used to change the X and Y location of the robot that is drawn as a square in the LabVIEW picture box.

Perform the following steps:

1. Using LabVIEW exercise #2 as a guide, copy the LabVIEW program given in the below two pictures. Notes for completing this exercise are as follows:

    a. Your robot's IP number is written on your robot car. The numbers written on our 12 robots range from 51 to 62. This number is the last number in your robot's IP 192.168.1.<yournumber>. If your robot has 55 written on it, your IP is 192.168.1.55. Also note that this new program has two while loops. One while loop is for sending eight floats to your robot and the other while loop is for receiving eight floats from your robot.

    b. The Send code / Event structure from LabVIEW exercise #2 is a good starting point for the Send of this exercise reminding you about the Timeout Event and Stop Event. In this case however, the Send event structure only sends data to the robot. The receiving of data happens in the second while loop. Eight floats, converted to text, are sent and I add these \FD and \FF characters as start and stop characters. Make sure to right click on these pink string constants and select "\ Display Code." Also make sure there is only one "\" back slash in the string constant. This tells LabVIEW to send the hexadecimal number 0xFD and 0xFF instead of the ASCII characters 'F' and 'D'.

    c. Make sure the Mechanical Action of both the Send and Stop buttons is set to "Switched When Pressed" by right clicking on the buttons.

    d. After creating the "Size of Rectangle" control, set it to 20 and then right click on it and select "Data Operations->Make Current Value Default." This will reload 20 in this variable the next time you open this program.

    e. To test your LabVIEW program you need to finish the remaining steps to setup both the red board and the ESP32.

2. In CCS create a new "LABstarter" project and rename it. Much of the code for communicating to and from LabVIEW, Nuttx, and the red board is given to you already in LABstarter. *If you are interested, the bulk of the communication code for the red board is in the UARTD receive interrupt function "RXDINT_recv_ready" in F28379dSerial.c*. The remaining code you need is given here:

    a. Cut and paste the following global variables towards the top of your project's C file. For now you can use these variables below. But in the future you may want to give the eight LV variables more descriptive names.

```
float printLV1 = 0;
float printLV2 = 0;
float printLV3 = 0;
float printLV4 = 0;
float printLV5 = 0;
float printLV6 = 0;
```

```
float printLV7 = 0;
float printLV8 = 0;
extern uint16_t NewLVData;
extern float fromLVvalues[LVNUM_TOFROM_FLOATS];
extern LVSendFloats_t DataToLabView;
extern char LVsenddata[LVNUM_TOFROM_FLOATS*4+2];
extern uint16_t newLinuxCommands;
extern float LinuxCommands[CMDNUM_FROM_FLOATS];
```

b. Then setup Timer 0's interrupt function to be call every 1 ms.

c. Copy the following code into Timer 0's interrupt function before the line of code that increments "numTimer0calls." Look as this code. "NewLVData" is a flag variable that is set for you when new data has been received from LabVIEW/Nuttx. When "NewLVData" is equal to one, copy the eight received floats into global variables you can use. Then also every 50 ms., this code sends eight floats up to Nuttx/LabVIEW. Look at the eight things we are sending. I use sin() and cos() and numTimer0calls to simulate X, Y coordinates. Then, so that we can see that the six other variables are being sent and received, I assigned them different fractions of "numTimer0calls." In the future you can change this code to upload the robot's actual X, Y coordinates and other data you want to send to LabVIEW.

```
if (NewLVData == 1) {
        NewLVData = 0;
        printLV1 = fromLVvalues[0];
        printLV2 = fromLVvalues[1];
        printLV3 = fromLVvalues[2];
        printLV4 = fromLVvalues[3];
        printLV5 = fromLVvalues[4];
        printLV6 = fromLVvalues[5];
        printLV7 = fromLVvalues[6];
        printLV8 = fromLVvalues[7];
}

if((numTimer0calls%50) == 0) {
        DataToLabView.floatData[0] = 3.0*sin(2*PI*.05*numTimer0calls*0.001);
        DataToLabView.floatData[1] = 3.0*cos(2*PI*.05*numTimer0calls*0.001);
        DataToLabView.floatData[2] = (float)numTimer0calls*.001;
        DataToLabView.floatData[3] = 2.0*((float)numTimer0calls)*.001;
        DataToLabView.floatData[4] = 3.0*((float)numTimer0calls)*.001;
        DataToLabView.floatData[5] = (float)numTimer0calls;
        DataToLabView.floatData[6] = (float)numTimer0calls*4.0;
        DataToLabView.floatData[7] = (float)numTimer0calls*5.0;
        LVsenddata[0] = '*'; // header for LVdata
        LVsenddata[1] = '$';
        for (int i=0;i<LVNUM_TOFROM_FLOATS*4;i++) {
                if (i%2==0) {
                        LVsenddata[i+2] = DataToLabView.rawData[i/2] & 0xFF;
                } else {
                        LVsenddata[i+2] = (DataToLabView.rawData[i/2]>>8) & 0xFF;
                }
        }
serial_sendSCID(&SerialD, LVsenddata, 4*LVNUM_TOFROM_FLOATS + 2);
}
```

d. In main()'s while(1) loop change your serial_printf statement so that it prints all eight values sent from LabVIEW.

```
serial_printf(&SerialA,"%.3f,%.3f,%.3f,%.3f,%.3f,%.3f,%.3f,%.3f\r\n",
printLV1,printLV2,printLV3,printLV4,printLV5,printLV6,printLV7,printLV8);
```

e. Build and download this code to your robot. Power the robot with 12V, 3A from the programmable power supply in lab. In CCS run your red board's code and open a Tera Term session to see what is printing. You should see the eight variable values sent from LabVIEW printed to Tera Term. They should be all zero as we have not setup the communication yet.

f. When the robot is powered on, the ESP32 board is also powered. It takes about 5 to 10 seconds for Nuttx to boot on the ESP32 board. After waiting 10 seconds, open a CMD prompt in Windows and test that Nuttx is ready by pinging your robot's IP:

   `ping 192.168.1.<your robot's IP number>` (i.e., `ping 192.168.1.55`)

   You should receive a response. Press "<CTRL> + C" to end the ping. If you do not receive a response with ping, check that you typed the correct IP. Also, you can try pressing the small "EN" button on the robot's ESP32 board, waiting 10 seconds, and trying again.

g. Then from the Windows CMD prompt connect to Nuttx over telnet by typing:

   `putty -telnet 192.168.1.<yourRobotsIPnumber>`

   This should give you a Nuttx Shell, nsh, prompt. At this prompt type:

   `tcpLVCOM`

   The tcpLVCOM application is now waiting for your LabVIEW program to connect.

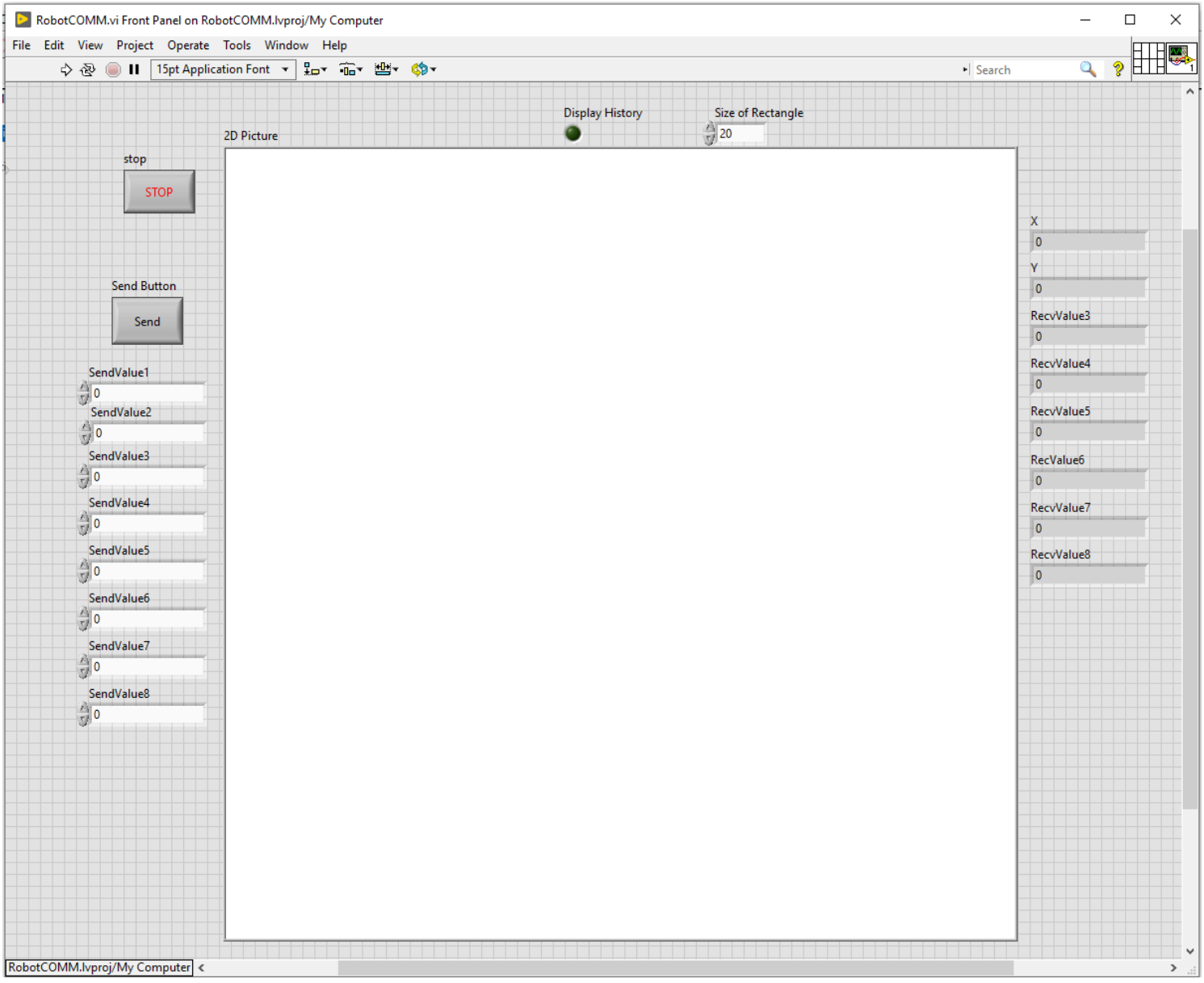
h. Make sure your red board's code is running in CCS and printing to Tera Term.

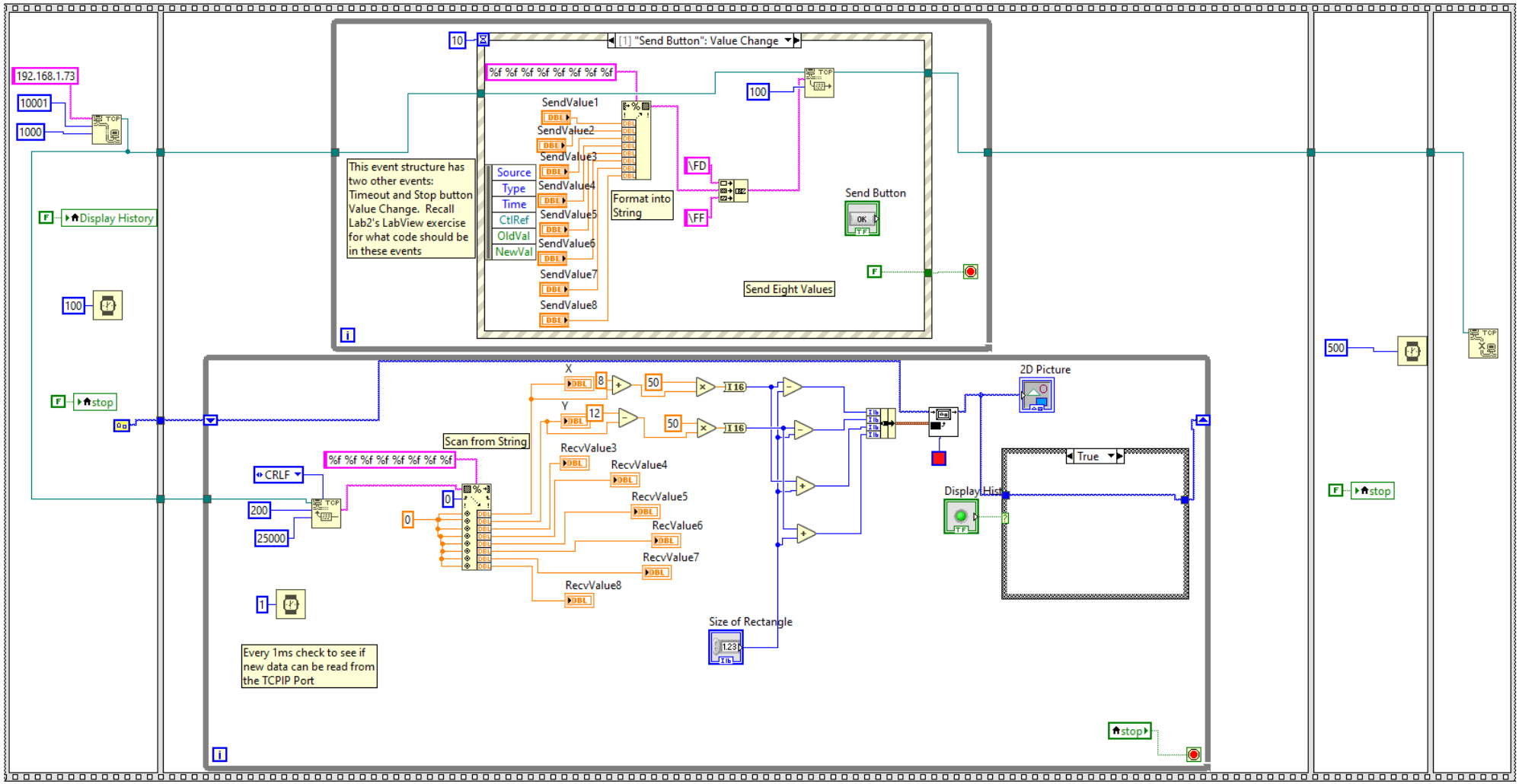
i. Run your LabVIEW program.

j. If everything is working, you should see your robot square moving in the Picture Box of LabVIEW. In the Nuttx shell, the values being received from the red board should be printing.

k. To test, download from LabVIEW to the robot. Type different floating point values in the eight send variables and press the "Send" button. You should see the printed values in Tera Term change to the values you typed in LabVIEW.

l. **Demo this working to your TA or Instructor** and/or continue to the last steps listed after the two below pictures.


RobotCOMM.vi Front Panel on RobotCOMM.lvproj/My Computer
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
Display History
Size of Rectangle
20
2D Picture
stop
STOP
Send Button
Send
SendValue1
0
SendValue2
0
SendValue3
0
SendValue4
0
SendValue5
0
SendValue6
0
SendValue7
0
SendValue8
0
X
0
Y
0
RecvValue3
0
RecvValue4
0
RecvValue5
0
RecvValue6
0
RecvValue7
0
RecvValue8
0
RobotCOMM.lvproj/My Computer


192.168.1.73
10001
1000
Display History
100
stop
10
[1] "Send Button": Value Change
%f %f %f %f %f %f %f %f
SendValue1
SendValue2
SendValue3
SendValue4
SendValue5
SendValue6
SendValue7
SendValue8
Source
Type
Time
CtlRef
OldVal
NewVal
This event structure has two other events: Timeout and Stop button Value Change. Recall Lab2's LabView exercise for what code should be in these events
Format into String
\FD
\FF
100
Send Button
OK
Send Eight Values
X
Y
8
12
50
50
I16
I16
2D Picture
True
Scan from String
%f %f %f %f %f %f %f %f
RecvValue3
RecvValue4
RecvValue5
RecvValue6
RecvValue7
RecvValue8
CRLF
200
25000
0
0
1
Every 1ms check to see if new data can be read from the TCPIP Port
Size of Rectangle
stop
500
stop

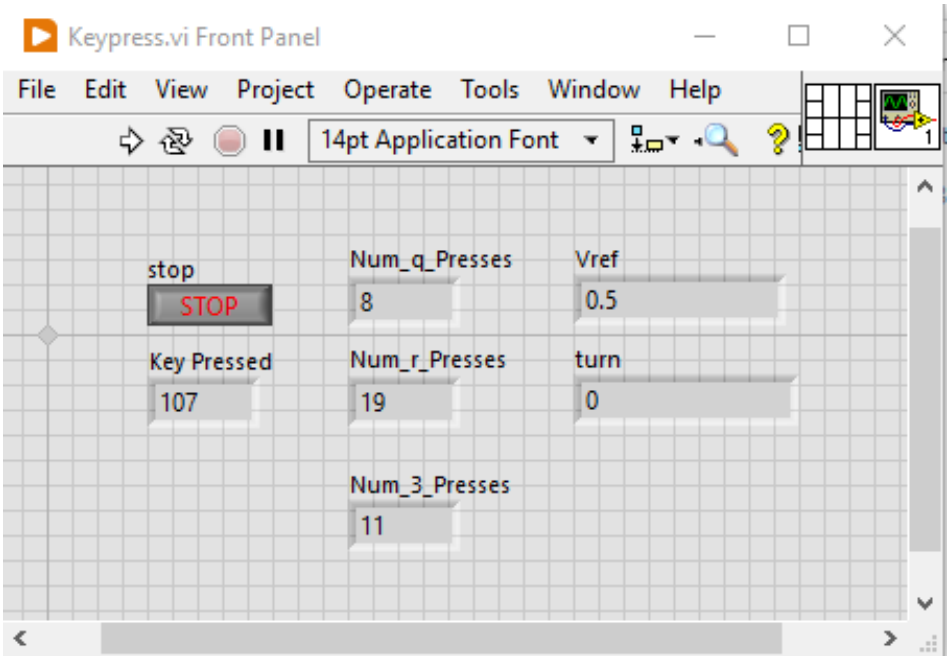
m. Add one more feature to this LabVIEW program that will make steering around your robot wirelessly a bit easier in Lab 6 and Lab 7.

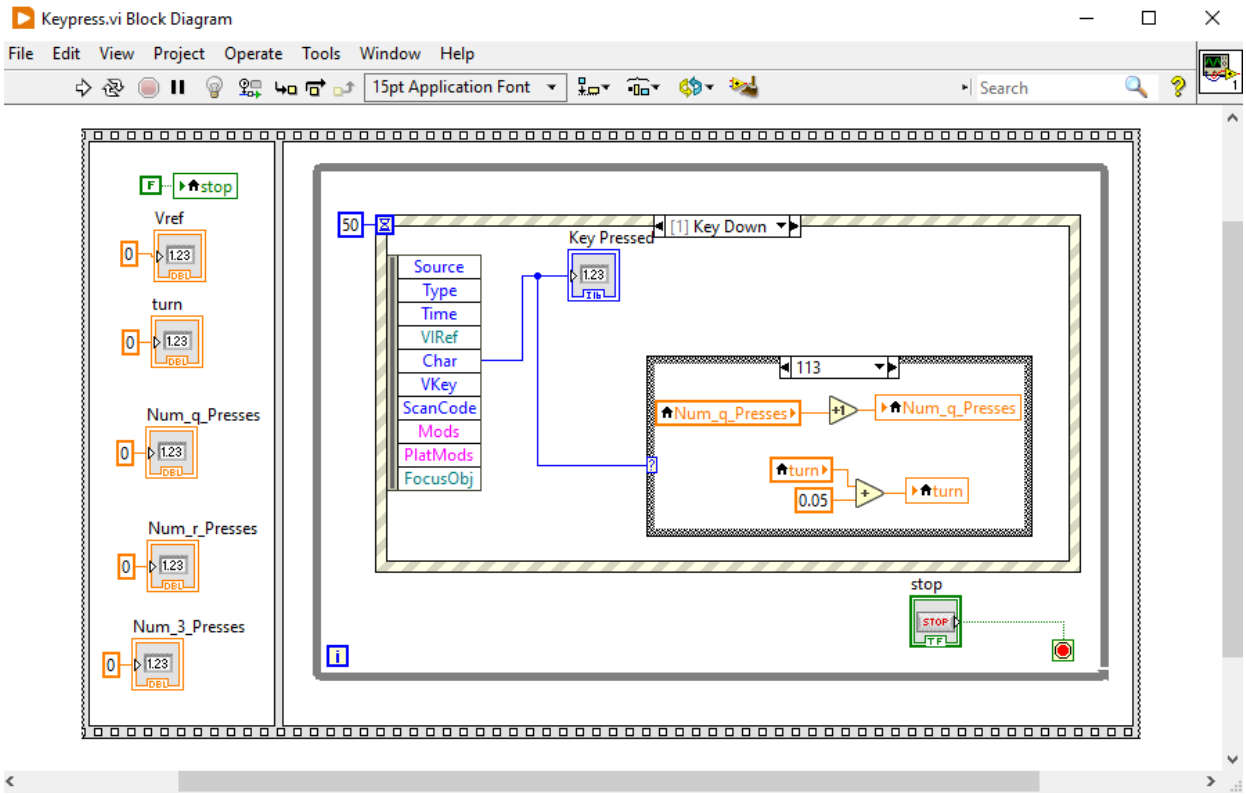
The Event structure has an event <This VI> → Key → Key Down. This event is run when your LabVIEW program has focus (mouse click the Front Panel View) and you press any key. Which key was pressed is given to you by the "Char" variable created by the event structure. So, using the below sample LabVIEW application, add to the above Event structure a Key Down event.

Copy most of what is in the "Send Button": Value Change event into this new Key Down event. BUT NOTE: you will have to use local variables to get the values of SendValue3 through SendValue8. Instead of using local variables for SendValue1 and SendValue2, write the value of "Vref" (Use a Local Variable block) to the first "Format Into String" input and "turn" to the second "Format Into String" input. You will probably have to make your Event structure a bit bigger to fit all the below code into your Key Down event.

Looking at the below example code, notice that when you press "q" (ASCII 113) the value of 0.05 is added to "turn". If "r" (ASCII 114) is pressed 0.05 is subtracted from "turn". If "3" is pressed a value of 0.1 is added to "Vref". If any other key is pressed, "0, Default", "turn" is set to 0 and "Vref" is set to 0.5. Add this last feature to your application and then **demo to your TA or Instructor** that your LabVIEW application receives values from the Red Board (through ESP32) and that both your Send Button event and Key Down event work to send new values to the Red Board.

Example Application to show you how to use the "<This VI> → Key → Key Down" Event. It is up to you if you want to build this small application first before you add the Key Down event to your Red Board interface App above.



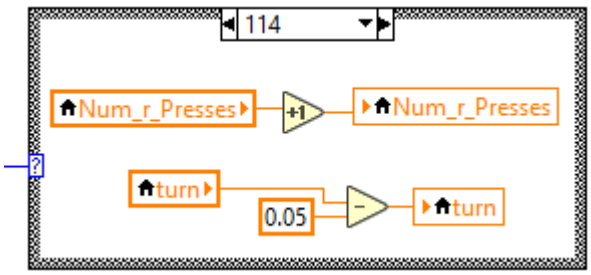


The three variables, "Num_q_Presses", "Num_r_Presses", and "Num_3_Presses" are not needed in your code. These variables are just incremented by one each time the Key Down event is called and that specific key was pressed.

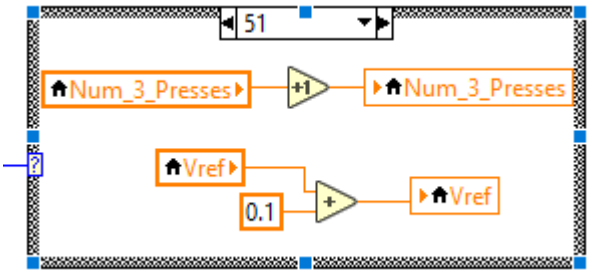
Notice how the Event Structure knows which key is pressed by looking at the event variable "Char". "Char" is an I16 variable, so it is the ASCII value for the key pressed: "q" = 113, "r" = 114, "3" = 51. Using a Case Structure with four items, the below code changes "Vref" and "turn". If "q" is pressed turn = turn + 0.05. If "r" is pressed turn = turn – 0.05. If "3" is pressed Vref = Vref + 0.1. If any other key is pressed, turn = 0, Vref = 0.5.

Key press case examples:

- Case for "r" being pressed:



- Case for "3" being pressed:



- Case for Any Other Key being pressed: