Project Title: Smart Display

Brief Project Summary

The main objective of the project is to design the server, client, spi interface, and the sensors so that different visuals can be displayed on the LED display. The server communicates with the client which either requests for sensor data or sends the file to be displayed. This file to be displayed is sent on the SPI interface which is then sent to a FPGA which interprets this information and displays the data. The sensors used are temperature, light, gps, velocity and camera. These sensors help decide what file is to be displayed.

List of Python Libraries used throughout the project:

spidev, time, socket, os, math, sys, serial, random, multiprocessing,w1thermsensor, RPi.GPIO,pygame,pygame.camera,datetime

List of Python scripts used:

1. newbinimp.py

2. spisender.py

3. clientonlaptopfnfin.py

4. sensor.py

5. main.py

# Script 1

This script is used by the spisender script to extract the rows of a file and add metadata each row.

The main functions used in this are extract and metdat. The extract function takes the number of pixels, the row that needs to be extracted, and the filename as the arguments and returns the extracted row in the variable extract(same name as the function). This extracted row, along with the amount of pixels, and the row number is sent to the metdat function which adds metadata to the row. The metadata contains the row number and amount of pixels. The metadata is 24 bits long(row and pixels taking 1.5 bytes each) since the fpga to which this data will eventually be sent to reads 12 bits at a time.

The functions and their inputs and outputs are:-

extract(): Input: fhandle,pix,row Output: extract(Type: list)

metdat():Input:datarow(basically extract),pix,row Output: finrow (Type: list)

Important variables used are:-

1)extract – this contains the extracted row which returns it back to the spisender script.

2)finrow- this contains the extracted row appended with metadata in the beginning

# Script 2

This script contains a function which is used for sending the data over the spi interface which is used by the main script.

The function is called spii. The function receives the name of the file from the main script. This file is opened and is sent to the functions extract and metdat in the newbinimp script. The order of the rows the spi requires for a 64 row file is 1,17,33,49,2,18.. so in this order the newbinimp functions returns the row data and stores it an a list called final. Finally final is converted into one big list of values in the required order called spidata and sent to the spi interface. 8 rows are sent at a time to the spi interface. Since the fpga’s frame rate requires each set of 8 rows to take up 4 ms time, it sends bunch of zeros, if the data of the rows is sent under 4 ms, till the 4 ms mark has reached. The enable is made active high and is high for when the row data is being sent and low for the rest of the time.

The functions and their inputs and outputs are:-

Spii(): Input: filename(received from the main script) Output: None

Important variables used are:-

1)spi – This holds the spi object

2)spi.max\_speed\_hz – This holds the clock frequency. For our project, we have put it at 15.6 Mhz

3)order – This is a list which holds the correct order of the row data we need to send.

4)spidata – This list contains the spidata that we will send over the interface in the required order.

# Script 3

The main function of this script is to send a file to the main script or request sensor data from the main script.

This script contains a main function called clientfrontera which has the inputs ip address,port, and option. The option value can be any value from 0-7(0-Camera access,1-Gps,2-Velocity,3-Temperature,4-Light,5-Updates all the values,6-Send file,7-Obtain log).Options 0-5 are used to obtain corresponding sensor data. Option 6 is used to send a file to the SPI interface. Option 7 is used to obtain the log.The The functions and their inputs and outputs are:-

sixbitbytestring(): Input:bin24 Output: p+q+r+s(bytes type string). This function is used to convert the number of packets into an encoded character string.

camera\_recevier(): Input: None Output: None. This function is used to receive camera data from the main script.

log\_receiver():Same as camera\_receiver except for log data

Important variables used are:-

1)client:- This hold the client socket object

2)filename – Holds the name of the file which will be sent to the main script running on the raspberry pi

3)totalpackets-This contains the number of packets the file which we will send has.

# Script 4

This script contains a class which holds all the sensor functions and the sensor variables which will be passed to the main script.

This script contains a class called Dsensor which holds the sensor variables and functions. When a Dsensor object is created, the variables dtemp,dlight,dimage,dvelocity are initialized to 0 or null values. The variables dlatitude,dlongitude,dtime and dcalcseconds are initialized to their real time and real position values. These 4 values are stored in a list called dvelcdata. The inputs for all the individual sensor functions are “self” since they are functions in a class and the outputs are None.

The important variables used are:-

1)dtemp – holds the temperature value

2)dlight – depending on the brightness, classifies as either dark,dim,normal and bright

3)dimage – list that holds the filenames of the 2 camera files

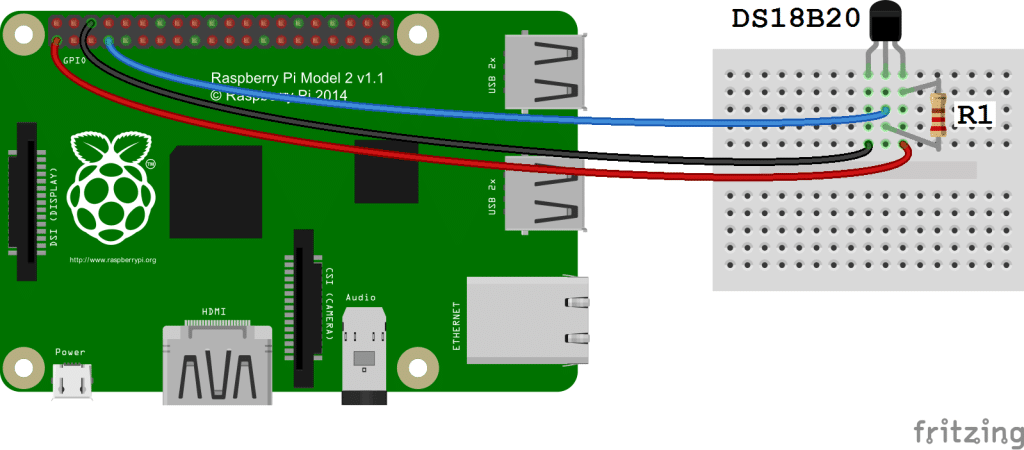
4)dlatitude,dlongitude,dtime, dcalcseconds – hold the latitude,longitude, ist time, and time in seconds respectively

5)dvelcdata – holds the variables dlatitude,dlongitude,dcalcseconds in a list and is used to calculate velocity data

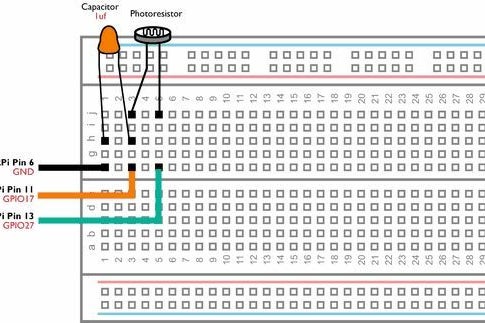
The functions in the class are:

1)Dtemperature(): This function is used to obtain the temperature. It is interfaced using 1-Wire protocol and is fairly straightforward. It obtains the information from the sensor and stores it in the variable dtemp.

1)The red wire(power/ of the sensor is connected to pin 1(3.3V) 2) The yellow wire(data) is connected to pin 7 or GPIO(4). This pin is used for 1-Wire interface. 3)The black wire(gnd) is connected to pin 6(gnd).



2) DLight(): Since the pi can only read digital values, a capacitor is used to find the resistance values of the LDR. As the capacitor gradually charges, the voltage that passes through the circuit and to the GPIO pin rises. Once the capacitor is charged to a certain point, it's voltage rises above 2 volts and the Raspberry Pi will sense that GPIO pin 13 is HIGH. If the resistance of the sensor increases, the capacitor will charge more slowly and the circuit will take more time to reach 2 volts. The script essentially times how long it takes for pin 13 to turn HIGH and then uses this measurement to calculate the resistance of the Photoresistor. Then according to the resistance, dlight is assigned either dark,dim,normal or bright.



Note: We have used a 4.7 uF capacitor. Resistance values will change for different capacitors.

3) Dgps(): This function is used to obtain the gps data. The gps data is read from the UART receiver port of the raspberry pi which is interfaced with the gps module(transmitter pin). It then stores the latitude and longitude values in dlatitude and dlongitude respectively.

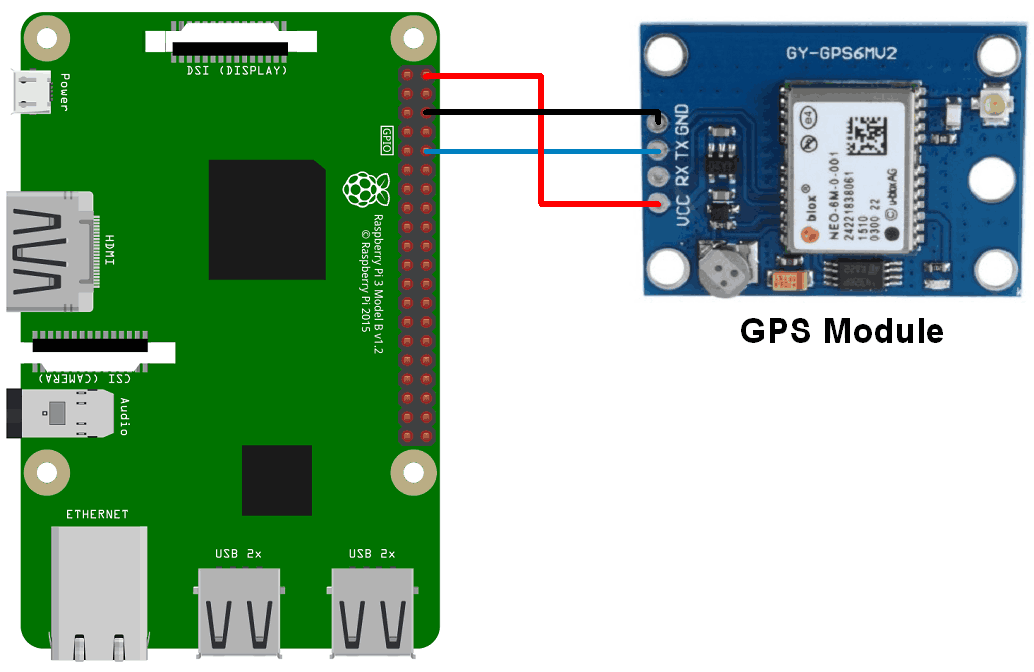
The functions used within this are:

ist\_and\_calctime(): Input(nmeatime) Output: isttime,timeinseconds(Type: Integer). This function is called by the GPS\_Info function and takes the NMEA time received from the gps module and converts it into Indian standard time. It also converts the current time into time in seconds and returns both the IST and time in seconds.

GPS\_Info(): Input: NMEA\_buff Output: lat\_in\_degrees,long\_in\_degrees(Type: Integer). The main purpose of this function is used to extract the gps data from the data received from the GPS module. The GPS\_Info function calls the ist\_and\_calctime function and stores the isttime in variable dtime and timeinseconds in variable dcalcseconds.

convert\_to\_degrees(): Input:raw\_value Output: position(Type: Integer). This is used to convert the latitude and longitude values to degrees.

1)The vcc pin(5v) of the sensor is connected to pin 2(5 V) 2)The TX pin of the sensor is connected to pin 10(UART receiver). 3)The gnd pin is connected to pin 6(gnd).



4) Dvelocity(): This function is used to calculate the velocity and stores it in the variable list called dvelocity. This function initially calls the Dgps function to obtain the current position and uses these values to find the distance between the current position and the previous position stored in dvelcdata. The variable dvelocity contains a list of 3 velocity values. The third value is the most recent value or the updated value. Once the value is updated, it takes the current position values and stores it in dvelcdata.

The functions used within this are:-

distbetween():Input:lat1,lon1,lat2,lon2 Output:distance(Type:Integer). This takes two pairs of coordinates and calculates the distance between them

5)DCamera(): The function is fairly straightforward. It creates a camera object, takes a picture, stores the image in the same folder as the script. The name of the image stored is “img\_date time”.

6) Update(): This function calls the Dtemperature, Dlight, and Dcamera functions and updates the sensor data.

# Script 5

This is the main script which uses multiprocessing to divide the three required tasks(server on pi, sensor, and spi) on three different processors.

This script implements multiprocessing and uses queues to communicate with the three different processes. The main function here is called f() which takes the three queues and the Dsensor object d1 as input and implements the three different tasks concurrently.

The first process is the server on pi process which communicates with the client. According to the data requested by the client, the server performs the required operation. If sensor data is asked, the server puts the request on the first queue “que”. Once the sensor data is obtained from que2, it sends the data back to the client. If the client wants to send the file, the server receives the file along with the name of the file and the number of packet and puts the filename on que3.

The functions in this are: sixbitbytestring(): Same as the one implemented in client. camerasender() which is used to send the camera sensor data. It is basically the counterpart of the camerareceiver() function in the client script.

The second process is basically the main process which communicates with the spi interface and the server. This process updates the gps and velocity values every 5 seconds so that we always have the real time values for both. This process also calls the respective sensor functions and puts their data on que2 to communicate with the server. It also puts the filename received from the client to the server on que3.

The third process calls the spii() function from the spisender script. The filename is taken from que3 which is then put as input to the spii function.

Important variables and functions used in this are:-

1)q1-server to main queue

2)q2-main to server queue

3)q3- main to SPI queue

4) d1- Dsensor object

5)logeditor():Input loginfo Output: None. This function appends data to the log file.