Final Project

MonitoringApp.py

ViewData.py

Sensors and accessories to interface (BCM pins)

- RGB LED (output) (Red = 5, Green = 6, Blue = 13)
- Regular LED called “LED” from now on (output) (pin 14)
- Buzzer (PWM output) (pin 18)
- Ultrasonic Sensor (input) distance in cm (ECHO = 24, trig = 25)
- Potentiometer (input) (Pot): Record the position of the pot in Percentage (SPI bus for MCP3008)
    - Option 1: without MCP3008: Use pins 7, 8 , 9, 10, 11 etc. based on number of bits you are using
    - Option 2: With MCP3008 : MCP3008 (use SPI0 with CE0)
- Push button (input) (pin 16)

Note: The pins are BCM pins.

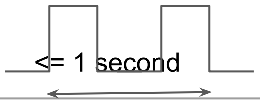
Record Data from sensors at the following sampling rates

Sensor	***Sampling rate***	***What to record?***
Potentiometer	0.5 second (10 % error margin)	Percentage turned (position) (2 decimal places)
Sonic (Distance)	0.1 seconds (10 % error margin)	cm (2 decimal places)
Time	For every sensor sample recorded	Current date and time (upto ms resolution)
Button	You can choose depending on the functionality explained further to experience non delay response	No need to record data. Use for mode change

The sampling rate should be followed within 10% error margin. So for sonic sensor samples should show up every 90 to 110 ms and for potentiometer, they should be every 450 to 550 ms.

Modes of operation

Push button will be used to select three “modes of operation” (MOP). “MS: Monitor System”, “RDM: Record Data & Monitor”, “ORD: Only Record Data”

MOP	Invoked by / Stopped by	Mode Indication	operation
MS	When program runs first. Or when the button is pushed once (single click), invoke the mode on the **button release**.	LED OFF	Explained in next slides
RDM	Button is pressed twice within 1 sec. Like a double click. Invoke the mode **on the second release of the button**. 	LED ON	Records the data from the sensors stated in slides 3 at the rate stated. Monitor System still works during record data mode.
ORD	Button pressed and held for >= 2 sec. Invoke as soon as 2 seconds are passed. User may release the button after 2 seconds. ORD should still be operational.	LED blinking (2 Hz)	Only recording, No Monitor System
OFF	Keyboard interrupt during any mode.	LED OFF	Exit the program. Stop all operation.

MS Mode: Monitor System (Mode indication LED Off) (40 points)

- Distance and sound : 20 points
  - Distance should be checked in cm every 0.1 seconds.
  - Buzzer frequency (tone) should correspond to the distance of the object. Increase the frequency of the tone starting from 100 Hz to 2 KHz as the object comes closer to the sensor. Map the distance from 4 cm to 20 cm to frequency 2 KHz to 100 Hz.
  - Distance > 20 cm, no sound. Distance < 4 cm, beep with 2 KHz tone at 1 Hz
  - ***Print the distance in cm and generated frequency to the console with right units***
- Potentiometer and RGB LED: 20 points
  - Potentiometer % rotation should be monitored in percentage every 0.5 seconds. RGB LED color (hue: R+G+B) should correspond to the potentiometer position.
  - Use human visible spectrum(https://en.wikipedia.org/wiki/Visible_spectrum) to map the RGB color to the distance: From Red to Violet corresponding to one end of the pot to the to other end. Hint: You should be able to find out RGB value of visible spectrum colors and map it linearly.)
  - ***Print the potentiometer % and RGB value which you are generating on the console.***

Record Data & Monitor: Mode LED on(20 points)

- Record data with specifications stated on slide 3.
- Data from the sonic and potentiometer will be recorded with the exact time it's sampled. Time should be system time (date, time to the ms)
- Data may be stored in a text or csv format. File should be easily opened from the folder by double clicking on it and data should be manually verifiable including the time. (Include a header)
- Data being recorded should be indicated by messages on the console as "Recording data….<filename> " and "stopped recording…<filename>" in the beginning and at the end of the recording mode session.
- ***Save the file at "/home/pi/Documents/" folder.***
- Filename for the data recording should be unique for each recording session. (You can use time and date into the file name.)
- System monitoring operation should be operational during this mode as described in the previous slide.

ORD Mode: Only Record Data (LED blinking at 2 Hz) (10 pts)

- System monitor operation should stop **(RGB LED off, Buzzer off)** in ORD mode.
- As far as recording data is concerned , everything applies exactly as in RDM mode.

ViewData.py: Reading records and plotting graphs (20 points)

- ViewData.py needs to run on separate machine/laptop/computer
- This will have following command line arguments:
  - Your Pi IP address (string)
  - Whether to plot the data or not ("0": simply transfer the file/s, "1": transfer & plot the data)
  - Filename: This argument can be used to decide which file should be transferred from the pi (***/home/pi/Documents/***) to your machine (**./DataFiles**) as well as during plotting which file record data should be plotted. When filename is not mentioned during transfer option all the recorded files should be transferred. But during plotting mode the filename option is necessary to decide which data to be plotted. During transfer operation if the file already exists then no need to overwrite it. Simply let user know that file already exists. (Remember that each file has unique name)
  - Find out which OS your code is running on and choose to have the local file path with correct slash sign. (./DataFiles vs .\DataFiles\)
  - During plotting operation if the file does not exist, transfer the file first and then plot it. Print appropriate messages to the console for the user to know what the system is doing. If the file does not exist on the pi then print the warning message.
  - ***While plotting create 2 subplots, one for potentiometer data and another for distance data.***

Example of how a program will run

- Python ViewData.py 192.168.1.200 0 data200_0412.txt : Transfer file data200_0412.txt from pi to laptop
- Python ViewData.py 192.168.1.200 0 : Transfer all data files from ***<u>/home/pi/Documents/</u>*** folder to your machine
- Python ViewData.py 192.168.1.200 1 data100_0412.txt :
    - If the file exists, plot all sensor data from “data100_0412.txt” using subplots for each sensor data
    - If the file does not exist, transfer it from ***<u>/home/pi/Documents/</u>*** and then plot the data.

Hints and pointers:

- How to transfer data from pi to your machine
  - Assume that your pi and the machine are on the same network.
  - Find out the IP of the pi manually using “ifconfig”. Use this address as an argument in your viewdata.py code when you run it from command prompt.
  - The ***/home/pi/Documents/*** path will be used in the ViewData.py code (hardcoded) so that the code knows which folder to transfer the file from.
  - You may not need client connection to transfer the files. You may transfer the files using ssh and scp.
  - scp pi@<pi Ip address>:<path of the file> <local folder path> **E.g.** **scp pi@192.168.1.101:/home/pi/Documents/filename.csv .\DataFiles**
  - ***Do not hardcode the local folder path. It should be relative to your current folder (./DataFiles)***

Hints and pointers continued

- Plotting:
  - If the file format is not correct as expected because the user input a wrong file name, throw an error instead of going into infinite plotting loop.
  - It may take a long time for the plots to show up if the file is large, have an estimate and print it for the user if possible.
  - The data should be marked accurately as legibly. Make sure the y limits are adjusted to show all the data clearly.

Program sanity and efficiency (10 points)

- Do not leave the file open for the entire recording operation. Better option is keep a small amount of data in buffer, open the file, write the buffer (only append), close the file. This will avoid file crashing errors.
- Avoid printing unnecessary information to the console to get the sampling rates as accurate as possible.
- If the user uses keyboard interrupt (ctrl+z, ctrl+a or ctrl+x) to stop the program, the program should not crash. It should gracefully stop the operation (LED off, buzzer off, save the file if applicable) and print an exit message.
- Manage exceptions to minimize common user errors such as wrong arguments, wrong format, wrong file format, file doesn’t exist, no network connection.