IC231 Spring 2022 – Lab 2 – Measurement and Hardware timing

Srikanth Sugavanam, Erwin Fuhrer, ssrikanth@iitmandi.ac.in, erwin@iitmandi.ac.in

In this lab you will learn how to use the RPI I/O for measurements and how use an approach that enables high time precision

Learning outcomes

In this lab, you will learn how to

- Use the GPIO pins to measure time intervals between events
- The accuracy and limitations of software-based timers
- The usage of hardware-based timings
- How to implement PWM (pulse-width modulation) to control the LED

Circuit connection instructions

Circuits as for last lab. Connect the positive leg of an LED through a current limiting resistor (470 Ω) to one of the GPIO pins, and the negative leg to a ground GPIO pin, as in the last lab. Connect the Oscilloscope such that you can measure the voltage at the GPIO pin.

Tasks

1. Rewrite the code from Lab 1, such that you have two control variables: `light_on` which allows you to control the duration for which the LED is actively blinking and "`frequency`" which allows you to control the duration T of a single period.
   Run the LED for 2 s and repeat the experiment with different blinking frequencies. Observe the jitter on the oscilloscope (adjust the horizontal window to analyse 10 periods). Repeat the experiment by varying the frequency and categorize your observations for the jitter strength (very strong, strong, medium, weak, not observable). Repeat the experiment while you play any game on the RPi in parallel.
   Important: Use the persistence mode on the oscilloscope (Manual page: 45)

Frequency	Jitter strength w/o game	Jitter strength with parallel game	Average T measured per oscilloscope
10			
400			
1000			
5000			
10000			
100000			

2. Change your code to run a signal with generated by pulse-width modulation (PWM). Use the command `gpio_pwm = gpio.PWM(channel,frequency)` to select the channel and frequency. In the next line use the command `gpio_pwm.start(DutyCycle)`. The duty cycle is a ratio value between 0 – 100 %. Run the code for 2 seconds with a frequency of 100 Hz and take a picture with your phone of the LED for the following conditions: DutyCycle =

[0,25,50,75,100]. Acquire the csv.-file from the Oscilloscope via USB (**(Manual page: 79 f. & 84 f.)**) for each setting and compare your observation on the LED with the PWM signal. Can you imagine any technical application?

3. Repeat Task I by using a DutyCycle of 50%. Do you observe any differences?

4. It is time to try the hardware timer. Before you start you need to initiate the pigpiod daemon as describe here: https://abyz.me.uk/rpi/pigpio/pigpiod.html . Start the daemon with a **sample rate of 4 µs.** (Hint: you can kill the process by using: `sudo killall pigpiod` and restart if required)
   Download from Moodle the code "lab2_task4.py". Complete the code by adding the missing snippets (You will find the information for the required inputs of the commands here: https://abyz.me.uk/rpi/pigpio/python.html). Which is the maximal frequency you can set? How many possible levels for dutycycle and frequency are there?
   Repeat task I and mark your observations.

5. Create a trigger signal with a single pulse with a duration of 10 µs. First use software trigger and then hardware trigger. Acquire the obtained trigger signal for both approaches using the oscilloscope. Analyse and compare your observations. (Save the .csv-files for your report.)

6. In the next step we want to measure the duration of the trigger pulse using a second GPIO pin as input pin. Connect GPIO pin 4 (BCM numbering) to measure the time duration of the trigger pulse using the Raspberry Pi.
   ➔ Download the code **"lab2_taks6.py"**. Measure the trigger duration by changing the value in line 18. Use 1s, 10 ms and 10 us. Does the code work? Justify!
      - **Important:** Check how we now use a GPIO pin as a measurement pin!

7. Next load the code **"lab2 _task7.py"**. Complete the code so you can run a trigger pulse of 10 us (check line 30). This code uses so-called event detections. What is the difference to the previous code with respect to the execution sequence? Execute the code ten times and note down the received pulse duration? Which values do you observe. Confirm with the oscilloscope. Can you make it more accurate so that the 10 us are repeatedly obtained with high accuracy? What needs to be changed?

8. Modify the previous code so that the computation of the pulse duration starts with immediately when the trigger pulse is finishing. (Hint: You need to replace the sleep function in line 33!)

Task completion criteria

1. Show your observations of Jitter strength
2. Explain your observation and show the LED images.
3. Explain your observations and make a statement if there are differences.
4. Explain possible differences to task I in the jitter.
5. Show the trigger pulse for hardware and software trigger
6. Make a statement if the code works as expected! Justify your answer.
7. Show the measurement of the trigger pulses. Explain your changes to precisely hit 10 us.
8. Show the revised code and the successful execution!

Pre-reading

- Hardware library in Raspberry Pi: The pigpio library
  - https://abyz.me.uk/rpi/pigpio/index.html
  - https://abyz.me.uk/rpi/pigpio/python.html
  - https://abyz.me.uk/rpi/pigpio/python.html#gpio_trigger

Circuit connection instructions:

M

Circuit connection instructions are identical to Lab 1. One more connection required to measure on channel 4.

Instructions

1. Wait for your TA to signal that the circuit connection is complete.
2. Write the program onto the Thonny IDE on the Raspberry Pi.
3. If you run into any issues, ask your TA/Instructor.
4. Generate for each task a new .py-file. In case something goes wrong you can go back to the previous working file.

Challenge exercise

Connect the green and the blue LED on two other pins. Create any color according to the RGB color chart (https://en.wikipedia.org/wiki/RGB_color_model). Can you sweep through the entire color space in 10 s? If not can you do a sparse sweep?