Day 2

Complementary filter

Joystick and Shared memory interface

Safety Limits

Today

- Complementary filter graphs– Milestone 1
- Interface main program with Joystick program and add safety shutoffs – Milestone 2

Common error from last week

- Include all required parts of report
  - Submit as pdf and cpp file on canvas
- void calibrate_imu(){

```
float sum_xg,sum_yg,sum_zg =0;
float sum_roll, sum_pitch = 0;
for (int i=0;i<=1000;i++){
      read_imu();
```

- Put computation of roll/pitch in read_imu()

- Make sure you clear the calibration variables before you calibrate (otherwise if you calibrate twice, the second time will be in error)

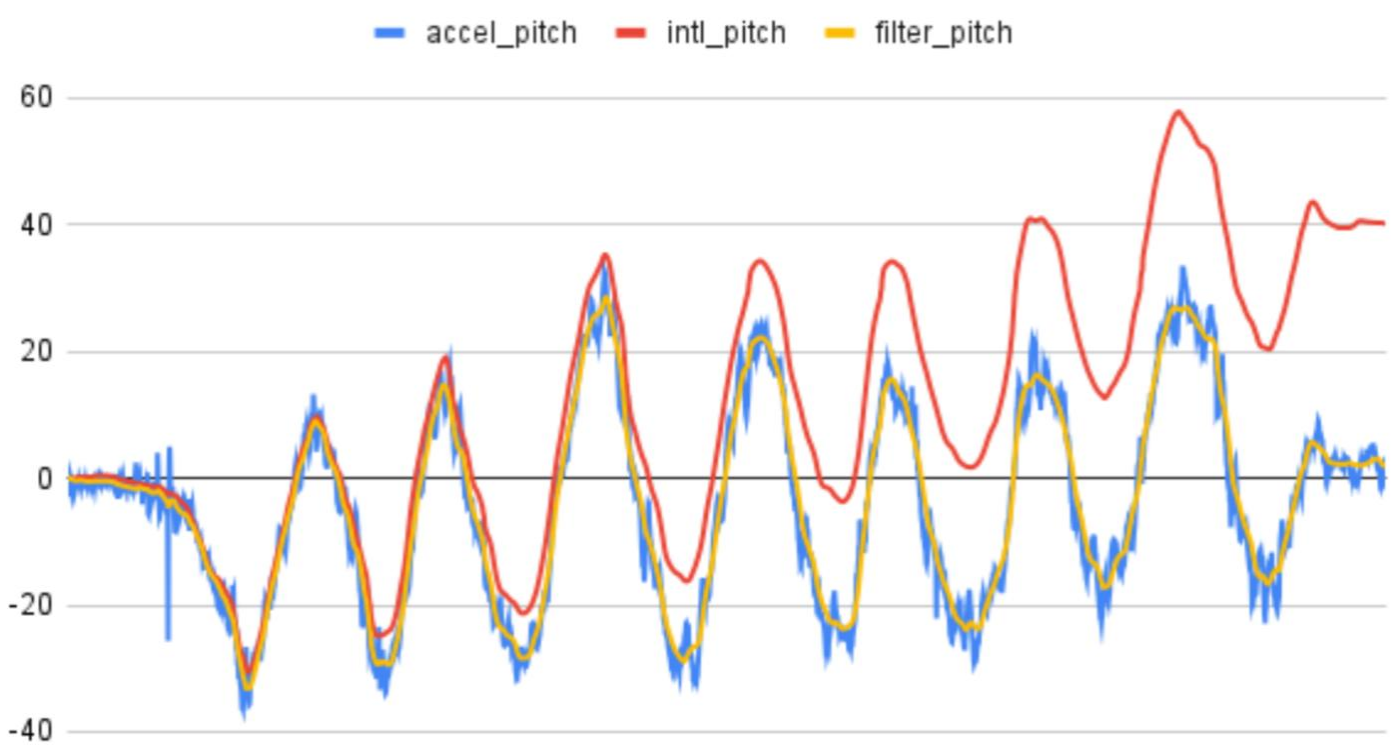
Pitch and Roll problems

- Use both accel and gyro
  - use the low noise of the gyro, but long term accuracy of the accelerometer.
  - High pass gyro + low pass accel
- Complementary filter, every time step t update:

  $Roll_t$=roll_accel*A+(1-A)*(roll_gyro_delta+$Roll_{t-1}$),

  Where A << 1 (try .02)
- Note: Make sure signs match, a + gyro rate should give a + increase in accel. angle.

Complementary Filter



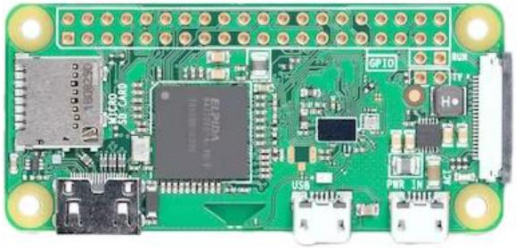
Milestone 1

- Create a graph showing: output of final comp. filter, roll angle computed by accel, and integrated gyro output for roll (all on 1 graph vs time).
  - Move the imu around gently (+- 20 degrees) during data capture
- Create a second graph for pitch

- Hint: easy way to do this is to print out data to screen using printf, copy to text file, import text file into excel and plot

Note: Be sure to download all files you made in class off the PI.. You'll need them for your weekly submissions

Joystick integration



User_code

Shared Memory

UDP_RX

WIFI

UDP_TX

Interfacing two programs

- Use udp_rx.cpp
  - Takes input from joystick and sends that to your program using shared memory.
  - Create a struct in student code, providing:

```
int key0;
int key1;
int key2;
int key3;
int pitch;
int roll;
int yaw;
int thrust;
int sequence_num;
```

- Desired behavior for student program:
  - If “B” button is pressed on joystick
    - Student program ends
  - If time since last new sequence_num > .35 seconds
    - Student program ends

- Note: you’ll need to have udp_rx.cpp running at the same time as your main program, open two ssh windows

Code updates

- See week2_updates.txt
- Udp_rx.cpp Udp_tx.py provided, no modification needed
- Change while(1) to while(run_program==1)
- After while loop, return 0;
- Be sure to update from shared memory before checking keyboard inputs
  - Joystick joystick_data=*shared_memory;
- void trap(int signal) function called when control+c pressed

Milestone 2

- Create function called safety_check() that..
- Student code ends in <.35 sec when any of the following conditions are met, printing out reason:
  - Any gyro rate > 300 degrees/sec
  - Roll angle > 45 or <-45
  - Pitch angle >45 or <-45
  - Joystick “B” is pressed
  - Joystick timeout
  - Control+c is pressed in student code
- Use #defines for limits so it is easy to modify later