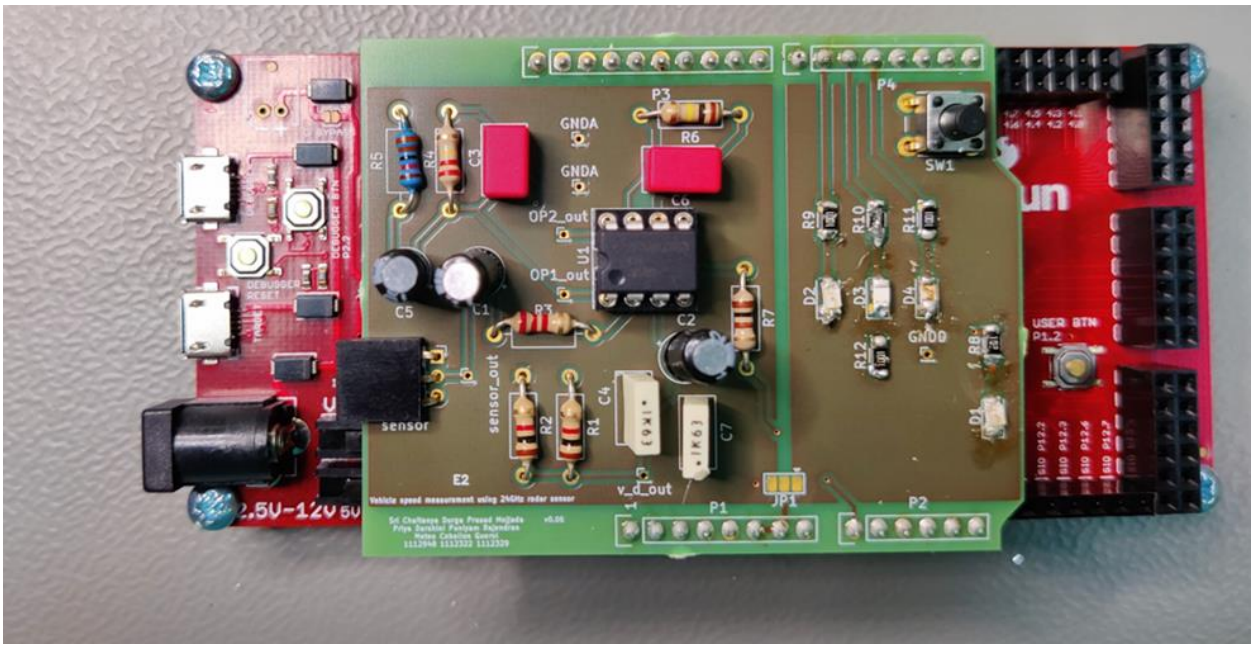
Speed detection using a 24GHz IMP 165 radar sensor



By:
Puniyam Rajendran Priyadarshini - 1112329
Mojjada Sri Chaitanya Durga Prasad - 1112322
Mateo Ceballos Querol - 1112948

28/06/2022

Table of contents

1. Requirements

The purpose of this project was to measure the speed of vehicles using the IMP165 24GHz radar sensor. It was intended to use the concept of doppler shift.

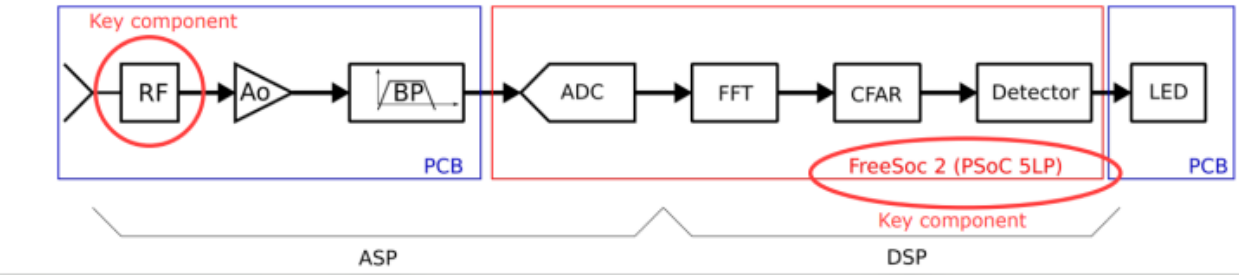


Figure 1: block diagram of project flow

KEY:

RF: Radar module
Ao: Amplification
BP: 4th order BP filter
ADC: Analogue to Digital Converter
FFT: Fast Fourier Transform
CFAR: Constant False Alarm Rate
Detector: Detect and track movements
LED: Light up LEDs on PCB to indicate distance

PCB:

RF module outputs analogue voltage which is then amplified and fed into the input of the BP filter.

FREESOC2:

ADC reads analog data which is converted to digital and sent to MATLAB via UART in which an FFT function is implemented. CFAR algorithm is implemented in MATLAB to increase probability of correct movement detections. Finally, when movement is detected, LED(s) turn on/off based on the movement.

2. Schematic

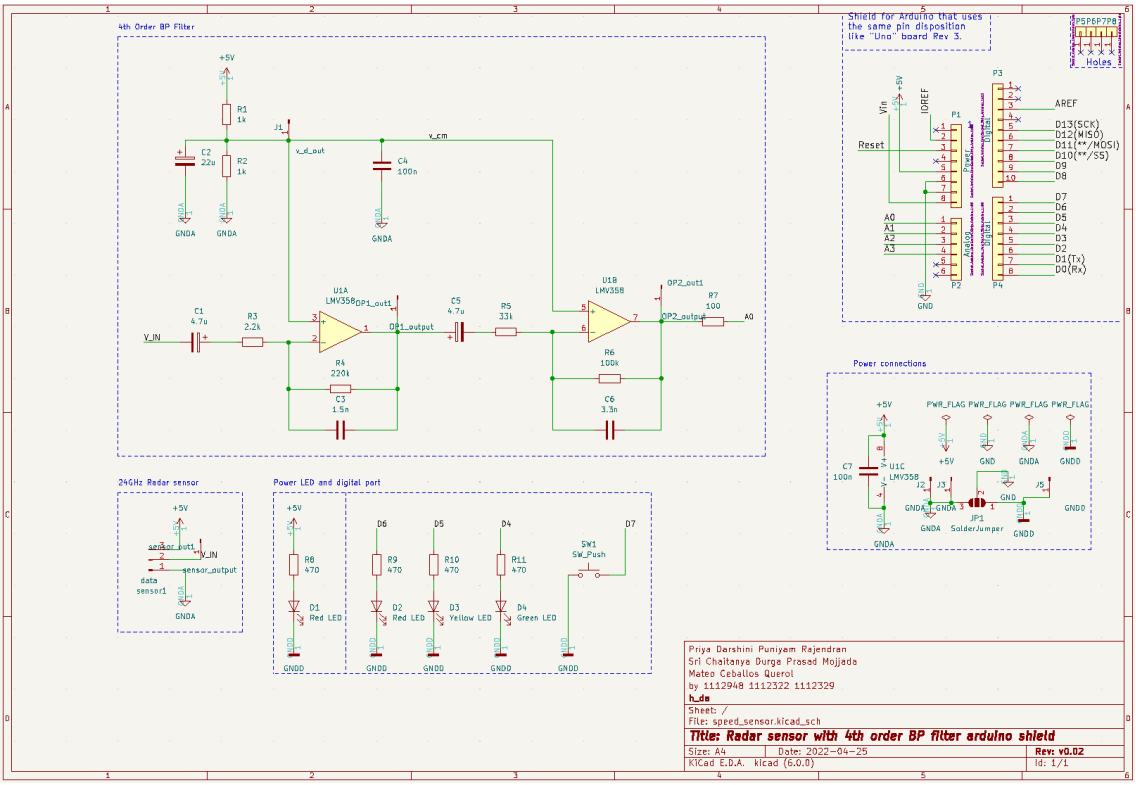


Figure 2: block diagram of project flow

- Arduino UNO R3 template
  - Arduino UNO R3 template was used for the PCB as per required
- 4th Order BP filter
  - To achieve the required amplification of 300, the resistor values used were 2.2k (R3) and 33k (R5). Voltage divider was meant to have an output of 2.5V therefore the resistor values were 1k and 1k (with an input of 5V). This was to raise the minimum threshold voltage from 0V to 2.5V. Stage 1 is a high pass filter followed by stage 2 which is a low pass filter. Three test-points were added: at the output of the voltage divider, at the output of stage 1 and at the output of stage 2 to monitor the different voltages.
- 24GHz Radar Sensor
  - A 3 pin header (90 degree horizontal) was used for mounting the sensor on the edge of the PCB. Another test point was added to be able to see the raw output of the sensor.

- Power LED and digital part

    - 3 LEDs (red, yellow, green) and a pulldown push button connected to different pins of the Arduino UNO R3 shield. The push button was used to start sampling of the data by the user.
- Power connections
    - There are 3 grounds, separated by a solder jumper. AGND for the BP filter and sensor then DGND for the LEDs and push-button. A global ground (GND) was connected to the Arduino UNO R3 template's GND

3. PCB

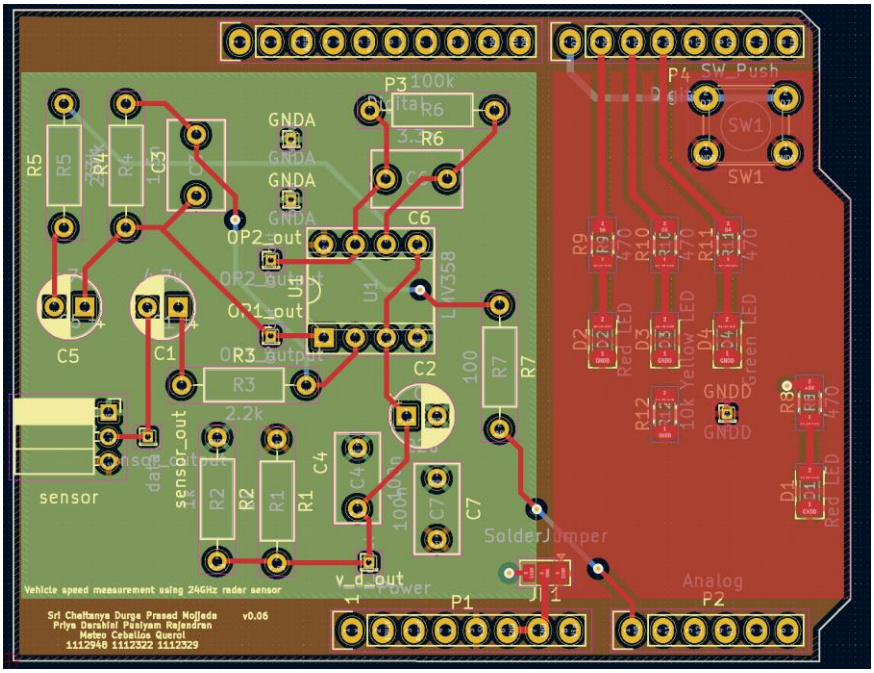


Figure 3: block diagram of project flow

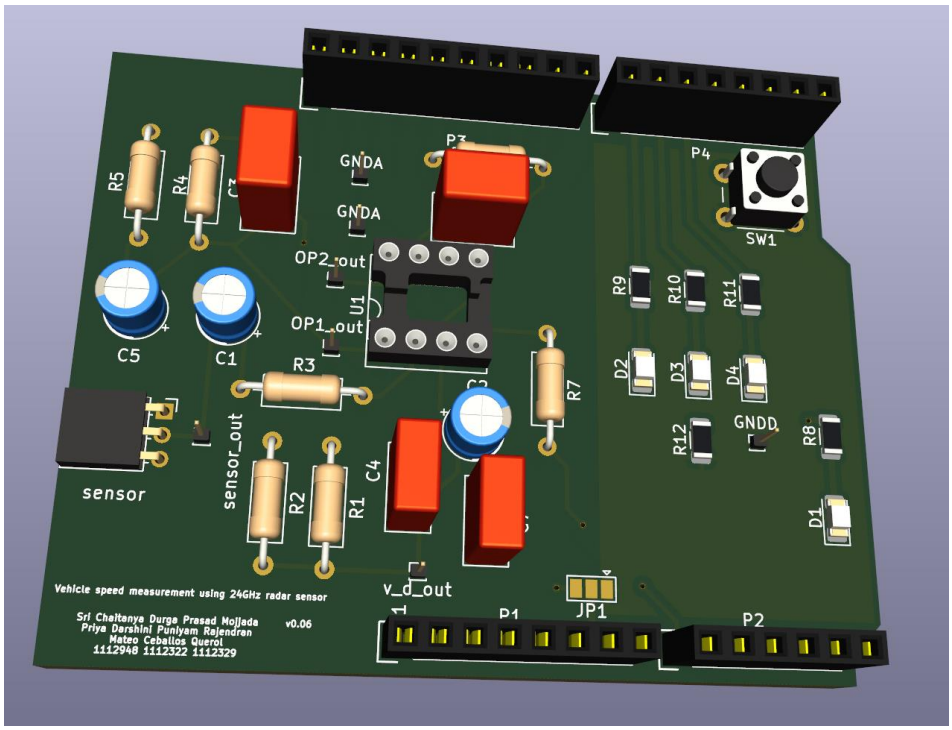


Figure 4: block diagram of project flow

4. Software

3 ISRs are used: UART, ADC and push-button. Software uses a state machine model in order to read and transfer the data into the MATLAB algorithm. The state machine is called endlessly.

STATE_IDLE:

This is the default state, it initializes the sample count to 0. The state machine will stay in this state until the button interrupt is triggered.

STATE_SAMPLING:

Reading the data of the ADC as 32 bits then typecasting to 16 bits while the character ‘s’ is not received via UART (sent by MATLAB). Once it is received, sampling is finished and the state machine goes into the STATE_UART_TRANSFER

STATE_UART_TRANSFER:

This state transfers the ADC array to the MATLAB algorithm via UART. The UART functions can only transfer 8 bits at a time, therefore the 16 bits was split into a HIGH byte and LOW byte and sent consecutively. Then the global sampling count variable is increased. Next, if ‘o’ is received and the count is less than 10, go back to STATE_SAMPLING and if 0 is received and count is equal to 10 then go to STATE_IDLE.

5. Testing

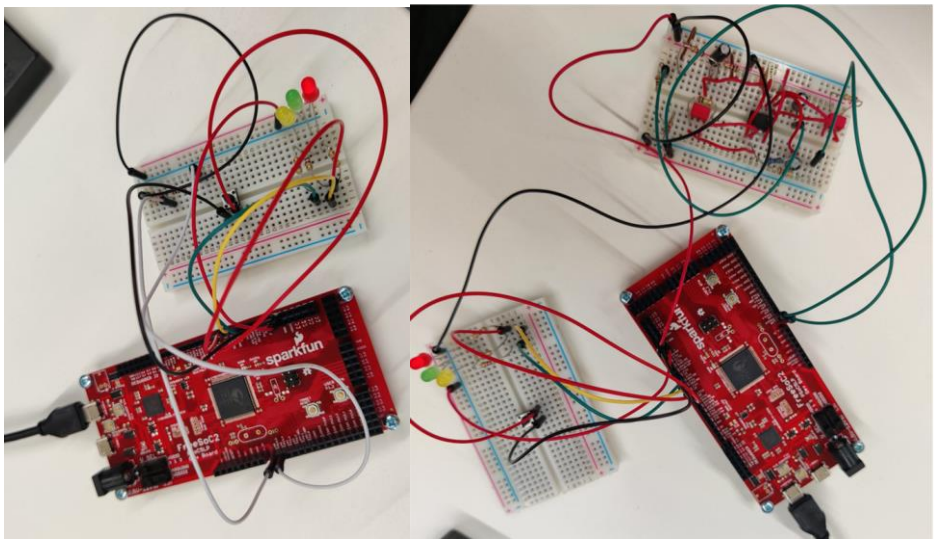


Figure 5a and 5b: Voltage divider test (left) and 4th order BP (right)

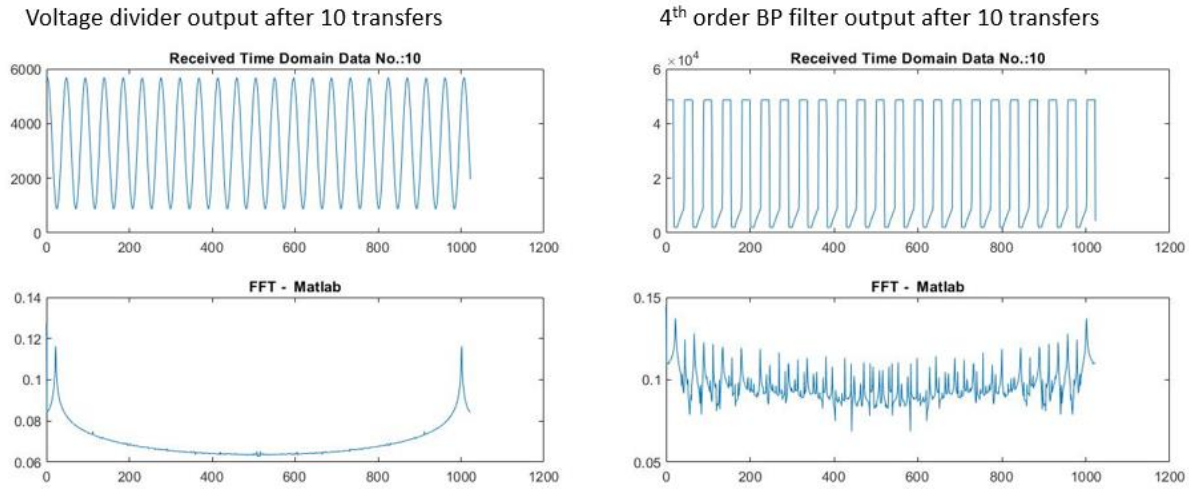


Figure 6a and 6b: MATLAB results after 10 samples with the voltage divider (left) and the 4th order BP (right)

Freesoc DAC and ADC are limited to 5V. In Figure 6b it is clipped because the ADC value post-amplification is beyond the maximum value of 5V even though the peaks can still be seen after FFT. However, with a lower amplification (voltage divider in figure 6a, it shows better results)

6. Final PCB

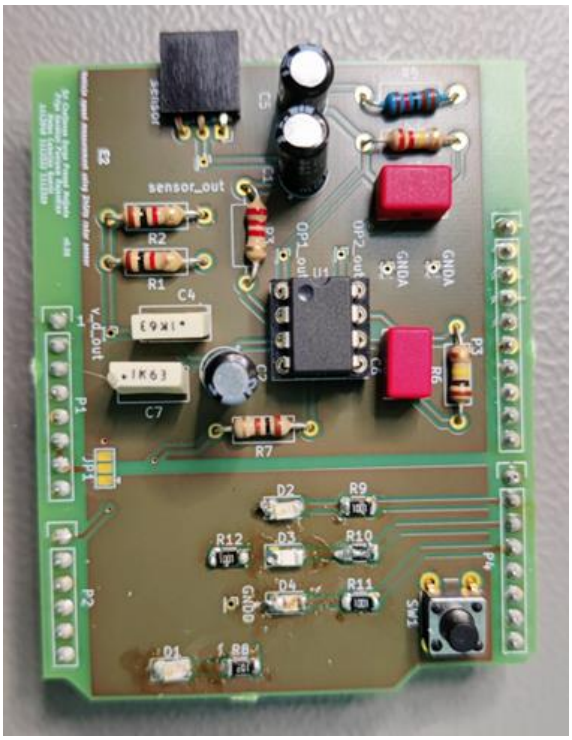


Figure 7: Final soldered PCB