Micro-Controller Lab Report

Ahmed Faheem Mostafa Maher

Qasim Haidari

March 4, 2019

Chapter 1

Introduction

RADAR stands for Radio Detection And Ranging and is a detection system that uses radio waves to determine the range, angle, or velocity of objects. It can be used to detect aircraft, ships, spacecraft, guided missiles, motor vehicles, weather formations, and terrain. Radars are now involved in various applications. It was initially used for target detection in military applications but nowadays it is used in tracking of aircrafts and vehicles and also in traffic control and space. In this project we will concentrate on motion and speed detection of objects using Radars.

The basic concept of Radar is sending an electromagnetic signal from a source which is radiating in the space. This signal will hit a moving object and then be reflected back to the source. In our case, the transmitter and receiver are on the same board of the radar device. The reflected signal from the object will be processed in the device and then finally sent to a microcontroller where the speed of the moving object is calculated using the Doppler effect.

The Doppler effect is the change of the frequency of an incident wave on a moving object in its reflected wave. The difference in frequencies is measured through a microcontroller to measure the speed of the moving object using the below formulas:

$$\Delta f = \frac{2 * v}{\lambda} \tag{1.1}$$

$$\lambda = \frac{c}{f_t} \tag{1.2}$$

where Δf is the difference between transmitted and received frequencies of the signal, λ is the wavelength, f_t is the frequency of the transmitted signal, c is the speed of light and v is the velocity of the object.

Chapter 2

Hardware Setup

2.1 Circuit Design and Components

In this project, we are using an input source for the signal generation and reception, an amplifier and filter stage, a processing unit, a display, two switches and three LEDs. Figure 2.1 shows the block diagram of the project.

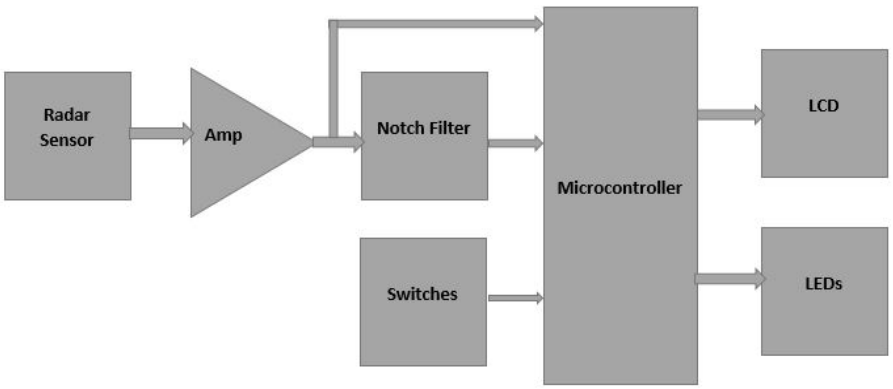

Figure 2.1: Block diagram of speed detection Radar

In the first stage, we use an IPM-165 Radar module which contains a full duplex patch antenna that transmits and receives signals in the K-band (24GHz). The reflected signal from the moving object has a slightly different frequency than that of the transmitted signal. This difference can be detected by the internal oscillator and the mixer inside the Radar module.

In the next stage, we use an amplifier to amplify the received signal for a more precise measurement and in order to increase the ability to detect objects far from the Radar module.

Due to possible additional noise in the environment which would affect the measurements negatively, we have also included a Notch filter that can suppress the noise at a certain frequency. We have included a filter with a 100 Hz frequency suppression as we might have neon light in the environment which would produce signals in this range. The drawback is that the respective speeds in the 100Hz frequency range will not be detected. So we are using both a filtered signal and a non-filtered signal in this project.

In the next stage we use the STM32L476 microcontroller. It converts the analog signal from the Radar module to a digital signal for processing. By processing the sampled digital signal, the microcontroller calculates the signal frequency and the speed of the object using the Doppler formula. The results for both filtered and non-filtered signals will be displayed on the LCD for user reading.

Two switches (one for switching between the filtered and non-filtered signal and one for freezing the maximum speed on the LCD) and three LEDs (for indicating the filtered, non-filtered and freeze states) are used for user requests and status indication.

2.2 Schematic and Hardware Considerations

There are some points that should be considered in the hardware design in addition to the previous main blocks. We will highlight them in this section as shown in the following schematics.

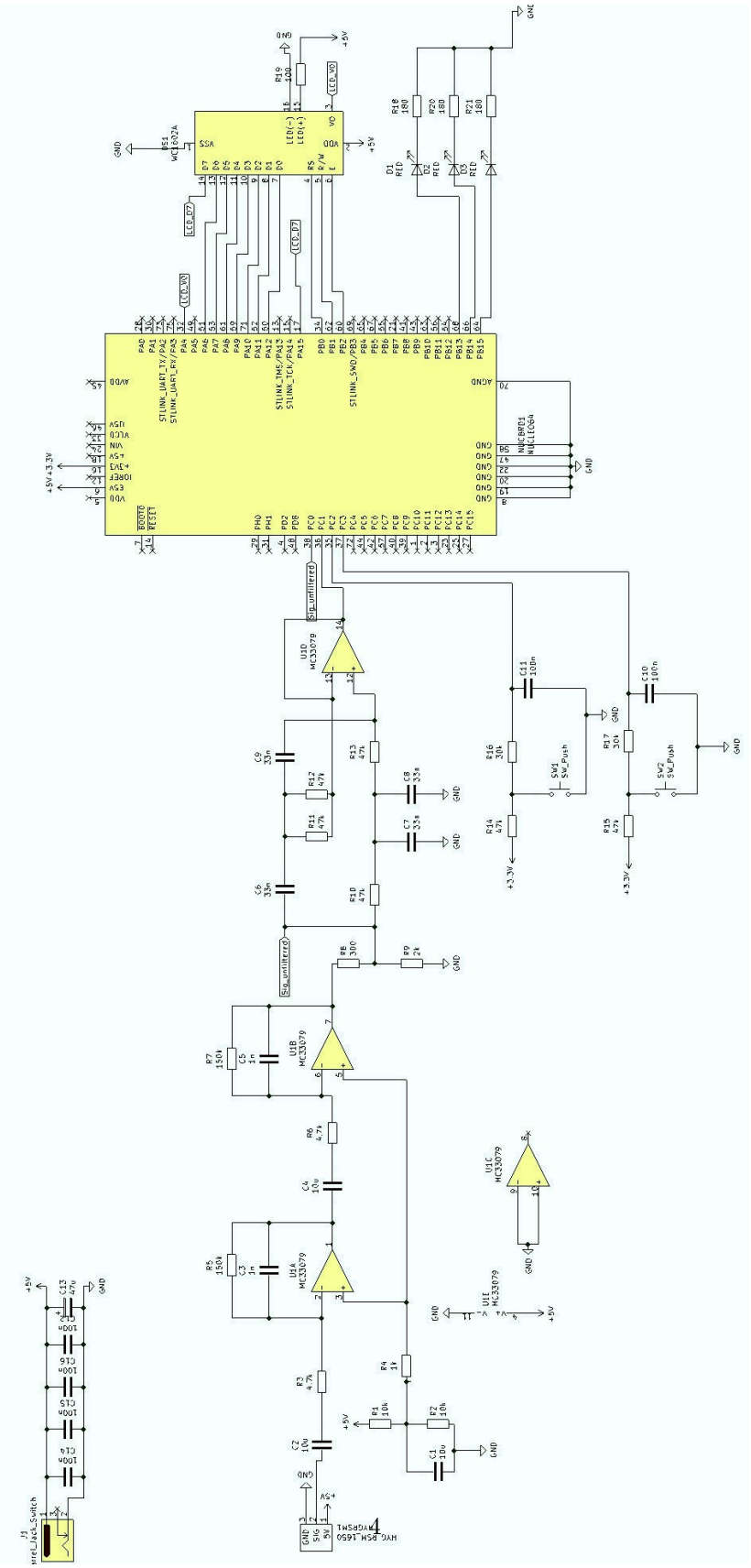

Figure 2.2: Schematic of the Radar circuit

Power Supply Decoupling Capacitors: It is highly recommended to connect capacitors to the input DC power supply to minimize the ripples that can be formed due to a non-ideal source. The capacitors C14 to C16 are used at the biasing of operational amplifier, Radar sensor and LCD and the capacitors C12 and C13 are connected to the input power supply. Figure 2.3 shows the schematic of decoupling capacitors.

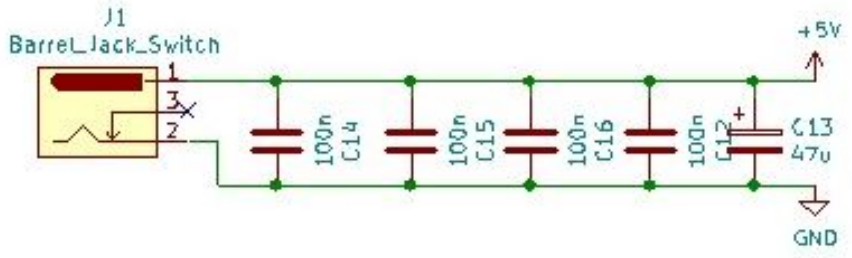

Figure 2.3: Decoupling capacitors for main power supply and power supplies of LCD, Radar sensor and operational amplifier

Operational Amplifier and Notch Filter: In our design, we used 3 Op-amps, and for size reduction we used a compact IC MC33079 which has 4 Op-amps. The first 2 Op-amps(U1A and U1B) are used in the amplifier circuit stage. The last Op-amp(U1D) is used as a voltage follower to isolate the effect of the load (microcontroller) on the voltage of the connected RC filter at the input of the Op-amp. The Notch filter is used to suppress the effect of noise at 100Hz frequency. Figure 2.4 shows the operational amplifier and the Notch filter with their connectivity to the microcontroller. The non-filtered signal is fed to pin PC0 and the filtered signal to pin PC1.

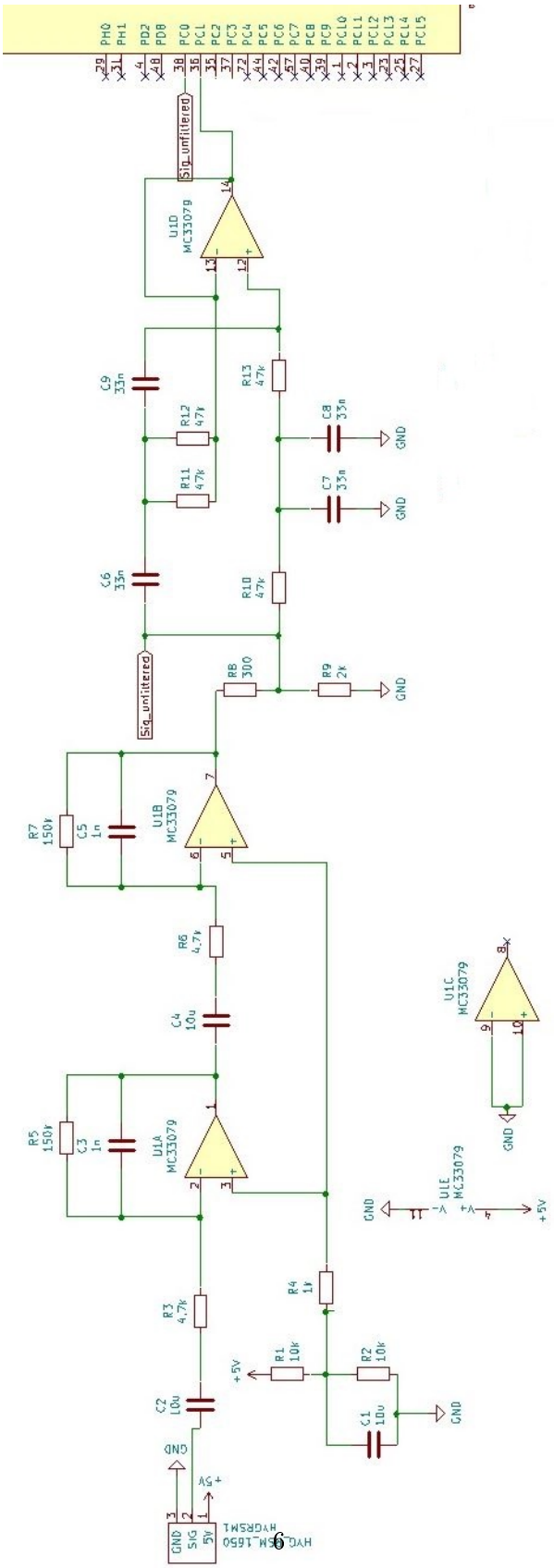

Figure 2.4: Operational amplifer and Notch filter connected to microcontroller

Switch Buttons: We used 2 switch buttons that should be connected to a debouncing circuit containing resistors and capacitors so that their bouncing when being pressed is canceled. Figure 2.5 shows their connectivity with the microcontroller. We used the formula

$$U_c(t) = U_0 e^{-t/\tau} \tag{2.1}$$

to calculate the time τ, and from it we can calculate the resistor value $R16 = R17 = 30K\Omega$ from

$$\tau = RC \tag{2.2}$$

Taking the bouncing time of the switches as $t\_bouncing = 0.85$ *(ms)* and the value of capacitor as $C = 100$ *(nF)*, we calculated

$$\tau >= 2.7(ms) \tag{2.3}$$

and this gave us

$$R >= 27(k\Omega) \tag{2.4}$$

We finally chose the smallest E24 resistor for *R16* and *R17* as $30(k\Omega)$.

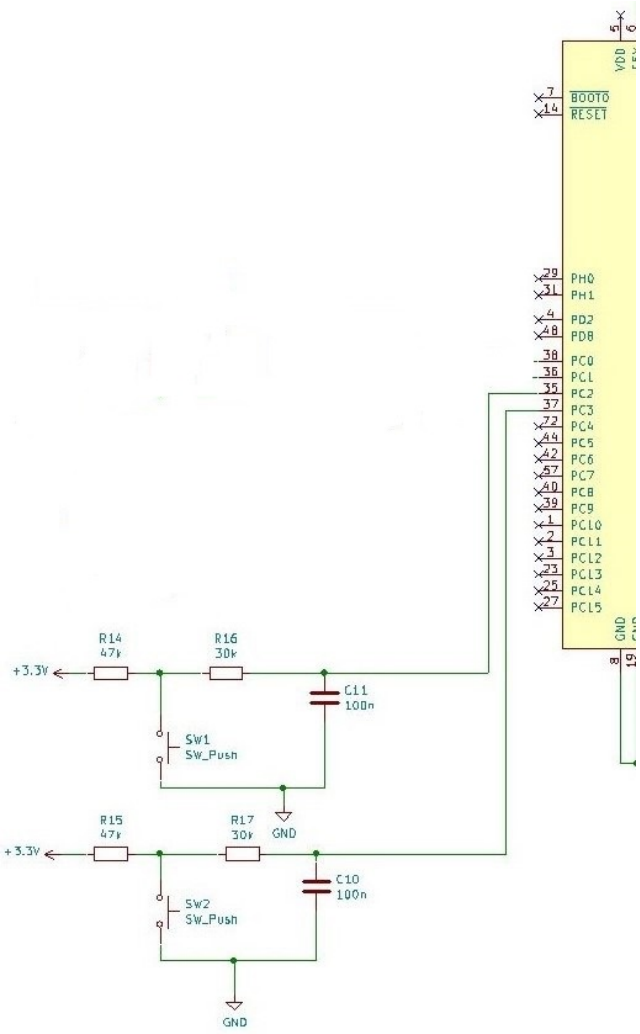

Figure 2.5: Two switches connected to microcontroller

LED Indicators: We used 3 red LEDs for status indication of the measuring process. We need a resistor at each LED to operate them at the desired intensity. According to the data sheet, the voltage drop was calculated using the Kirchoff's voltage law, and the resistors were calculated as R18=R20=R21=180Ω.

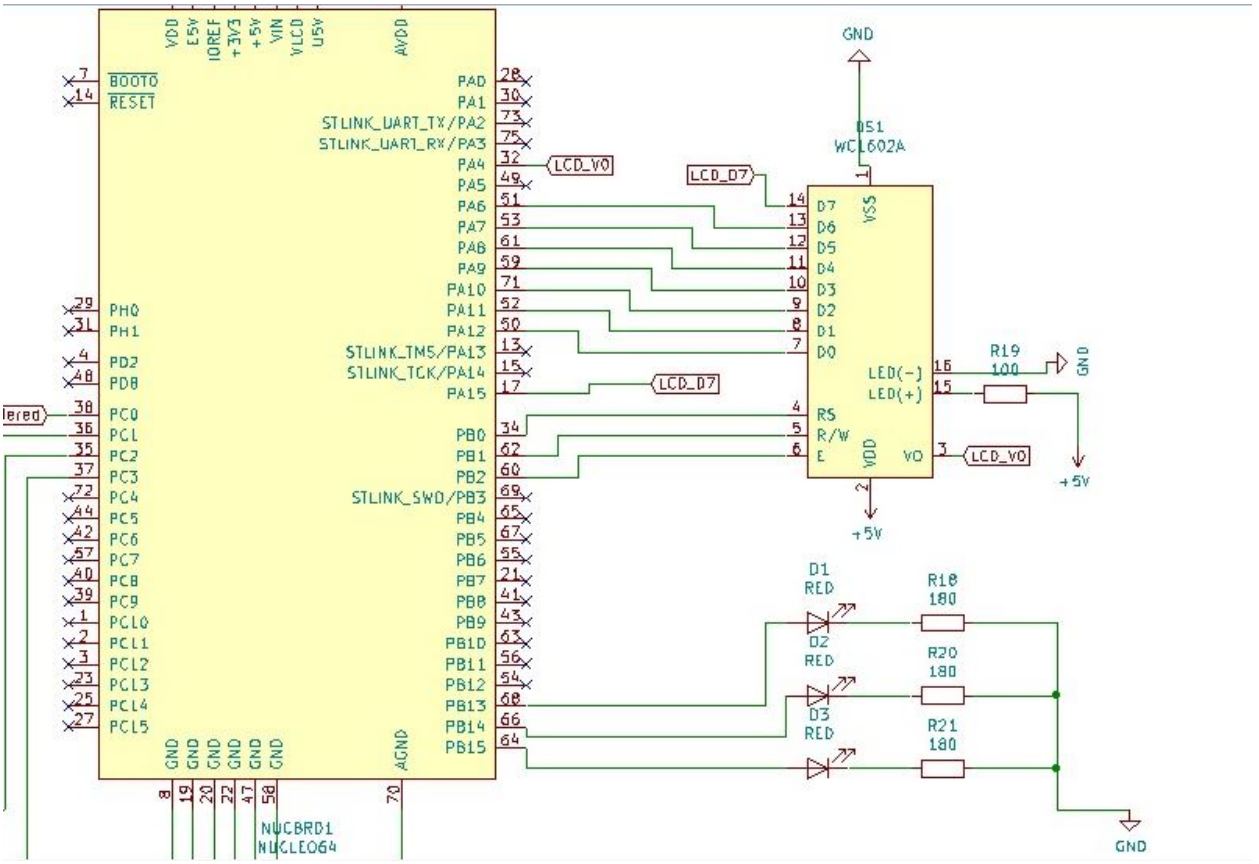

Figure 2.6: LCD and three LEDs connected to microcontroller

2.3 Operational Flowchart

Fig 2.3 represents the flowchart of the Radar operation.

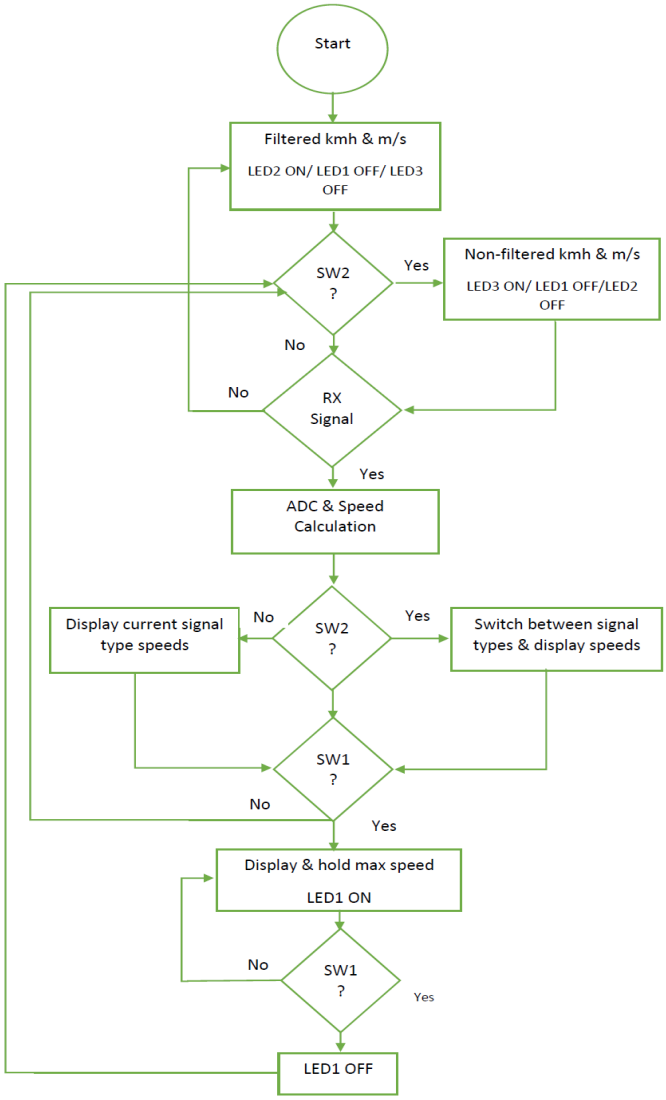

Figure 2.7: Flowchart of speed detection radar

In this project we have considered 2 scales for the calculation of speed: Kilometer per hour and meter per second, both are displayed on the screen. We can distinguish between filtered and non-filtered signals. The user can switch between them by using switch 2 and the respective speeds in kmh and m/s will be displayed on the LCD. Switch 1 is used to freeze the maximum speed which has been captured so far. LED1 will be ON to indicate the freeze state, and

LED2 and LED3 will be ON to show if the filtered signal or the non-filtered signal has been selected respectively.

Chapter 3

Software Concept

Interrupts: There are two switches (SW1 and SW2) which act as external interrupts. The following external interrupt callback function sends a specific flag to the queue depending on which one has been pressed.

```
/* USER CODE BEGIN 4 */
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin) {
    /* Prevent unused argument(s) compilation warning */
    //UNUSED(GPIO_Pin);
    int cIn = 0;

    BaseType_t HigherPriorityTaskWoken = pdFALSE;
    if (GPIO_Pin == SW1_Pin) {
        cIn = 1;
    } else if (GPIO_Pin == SW2_Pin) {
        cIn = 2;
    }

    xQueueSendToBackFromISR(defaultQueueHandle, &cIn, &HigherPriorityTaskWoken);
    if (HigherPriorityTaskWoken == pdTRUE) {
        taskYIELD()
    }
    /* NOTE: This function should not be modified, when the callback is needed,
     the HAL_GPIO_EXTI_Callback could be implemented in the user file
     */
}
/* USER CODE END 4 */
```

Figure 3.1: EXTI Callback function for switches 1 and 2; The function also shows the communication between ISR and the queue

In the main task, anytime the SW2 flag is received from the queue, a switch between measurements for filtered and non-filtered signal will happen. This is done by changing the value of a local variable (*state*) on each SW2-ISR to Task communication. Depending on the value of *state*, an appropriate channel of ADC which takes samples from either a filtered or a non-filtered signal is selected.

Global Variables: The following global variables with their respective initial values have been used in the program:

int final_count = 0; Indicates number of samples between two successive zeros in the signal
int timer_div = 0;
int sample = 0, *prevsample* = 0;
sample_count = 0; It is incremented each time a new sample is taken
freeze = 1;
double min_final_count = 30000;

Timer Settings for ADC Sampling: The timer parameters have been set as the following:

Prescaler = 1500;
Period = 1;
ClockDivision = 4;

Since the internal clock for the timer is 80MHz, these settings will make the final timer clock around 13333Hz. During each timer clock, ADC takes a sample and converts it to a digital value. This makes the sampling rate over 20 times the highest frequency (600 Hz) in the received signal (Nyquist criterion for sampling an analog signal with at least twice the highest frequency in the signal has been fulfilled).

Local Variables in the Default Task:

int cIn = 0;
int state = 1;
double half_period = 0, *doppler_f* = 0;
double v_kmh = 0, *v_ms* = 0;
char str[10];

Static Local Variable: The static local variable (*OFFSET*) is used in the timer service routine function to account for the offset value of the received signal.

Communication between Timer Service Routine Function and the Task:

Sending the flag *cIn* = 3 from the timer SR to the queue: It is sent to the queue whenever a new zero crossing is detected and the current value of *sample_count* is put into *final_count* indicating the number of samples for the previous half period of the signal. *sample_count* value is then set to zero to start counting the number of samples in a new zero-crossing to zero-crossing interval.

The flag *cIn* = 4 is sent to the queue whenever the global variable *timer_div* reaches 4000. *timer_div* is incremented any time a timer clock calls the timer service routine function in which a new sample is taken and counted. This flag is used in order not to refresh the contents on the LCD screen any time a new speed is about to be displayed.

Functions of Switches and LEDs:

SW2: Switching between filtered and non-filtered measurements
SW1: Freezing the max speed captured so far on LCD. The default state is not to freeze
LED2: Filtered signal has been selected
LED3: Non-filtered signal has been selected
LED1: Indicating the freeze state

Freezing the Max Speed on LCD: The global variable *min_final_count* = 30000 is used in combination with the global variable *freeze* to freeze the maximum speed which has been captured so far. Switching between freezing and non-freezing states is done by the flag $cIn = 1$ which is received in the default task from the EXTI Callback function.

Anytime a switch between filtered and non-filtered signal happens (through SW2), *min_final_count* is reset to 30000 so that the max speed for the new signal type will not be affected by the values of the last signal type. Also, if this switch happens in the freeze state, the speed values will be reset to zero so that the previous frozen max speeds will not stay on the LCD screen.

In the non-freeze state, whenever a new *final_count* is received (through timer SR to task communication), it is compared with the current *min_final_count* to update it for the max speed in the freeze state. In the freeze state, *min_final_count* is taken for measurements

Chapter 4

Conclusion

A Concluding Note on Frequency Calculation: When a signal is received from the Radar sensor, the calculation will be performed in the processor of the microcontroller. Since the highest signal frequency that is received is 600 Hz, the sampling rate according to the Nyquist criteria should be at least double this value. We have adjusted the timer parameters in such a way so that the timer service function is called 13333 times per second and within each call, ADC takes one sample. This makes the sampling rate over 20 times the highest frequency. Whenever one sample is taken, it is compared with its previous sample to detect two successive zero crossings and count the number of samples taken within these zero crossings. Since the time for taking one sample is known based on the sampling rate of the timer/ADC, the half period of the signal can be measured. During this process, an offset value and also a threshold value for canceling the noise have been considered. By knowing the period and frequency of the signal, the speed of the object will be measured using the Doppler formula.

The user can hold the reading on the LCD screen if switch 1 is pressed which will display the maximum speed calculated since the last reading. LED1 will be on to indicate the hold state. The user can release the hold state by pressing switch 1 again and the system will return again to wait for a new received signal.