ECGR 6189 / 8189 Wireless Sensor Networks: Project Report

Vehicle Distance Detection and Alarm

Table of Contents

1. **Project Objectives**
2. **Hardware**
3. **Software Utilized for Implementation**
4. **Hardware Review**
5. **Principle of Operation**
6. **Algorithm**
7. **Code Snippet**
8. **Design Complications**
9. **Execution**
10. **Future Scope**
11. **Conclusion**

ECGR 6189 / 8189 Wireless Sensor Networks: Project Report

Vehicle Distance Detection and Alarm

Project Objectives:

With the main aim of the project being Vehicle Distance Detection, displaying an alarm when an obstruction such as another vehicle draws in close, the same has been accomplished through an IR Sensor powered through the excitation voltage (5.0 V) of MDA-300 CA Sensor Board which is being powered by and mounted upon a MicaZ mote. With the IR Sensor output provided to one of the ADC Registers (A2) of the MDA-300 CA, the same is transmitted to the Base Station mote (MicaZ) for display of alarm along with speed changes and the new speed, if any. With the distance over which alarm is given out depending upon the speed provided, a higher speed shall result in alarm being given over a larger distance and vice-a-versa.

Hardware:

The following hardware was utilized in implementation of the Project:

1. 2 Mica Z motes (One as Base Station and one as a Sensor Node for the IR Sensor).
2. 1 MDA-300 CA Sensor Board Controller (to interface the IR Sensor with the mote).
3. 1 Sharp IR Sensor (GP2Y0A02YK) (for sensing).
4. 1 Programmer MIB 510 Board (to program the motes and enable Serial Listen).
5. 1 USB Extension Cable (for ease of programming the motes and implementing Serial Listen).
6. 2 resistors of 10 kilo Ohms (for MDA-300 CA Sensor Board).
7. 2 resistors of 2 kilo ohms and 17 kilo ohms (for implementing voltage divider).
8. Supply: Two AA batteries for a Mica Z mote. The second mote was powered through the USB Cable.

Software Utilized for Implementation:

The project was implemented though the Tiny Operating System (Tiny OS), an open source embedded, component-based Operating System and an operating environment to interface sensor motes. With the programming language utilized being the nesC i.e. Network Embedded Systems C, an event-driven, component-based programming language for the TinyOS platform, the programs were written and tested over Linux Ubuntu. While the command utilized to implement Serial Listen tool in C through the base Station mote is "*./seriallisten /dev/ttyUSB1 micaz*", the command utilized for compiling and installing the program over the MicaZ motes (base station and sensor mote) is "*make micaz install.2 mib510,/dev/ttyUSB0*" with the integer "2" representing the unique mote ID given to each mote.

Hardware Review:

MDA-300 CA Sensor Board

Providing Onboard sensor excitation voltage of 2.2 V, 3.3 V, 5 V, and low power mode along with 64K EEPROM for onboard sensor calibration data and 7 single-ended (each of which can tackle with voltage of 2.5 V)/ 4 differential ADC Channels and 4 precise differential ADC Channels, the MEMSIC MDA-300 CA Data Acquisition Board also houses Temperature and Humidity Sensors and can be interfaced with MicaZ and Mica2 motes for sensing applications.

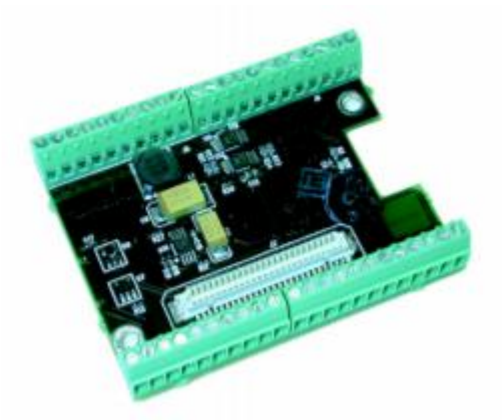

Figure 1: MDA-300 CA Data Acquisition Board.

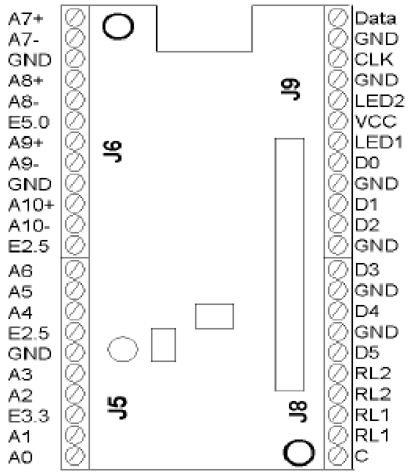

Figure 2: Pin Description of MDA-300 CA Data Acquisition Board.

Utilizing IR Sensor for the project, the same has been powered through E5.0 pin (pin 6) of the Board which provides an excitation voltage of 5V for the sensor. The output of the sensor is provided to the ADC Channel 2 i.e. A2 (pin 19) of the Board.

IR Distance Sensor

The Sharp wide-angle IR Sensor (GP2Y0A02YK), which has been utilized for the project, while taking 5 V as input voltage from the excitation pin of the MDA-300, provides an output voltage in the range of 3.0 V to 0.4 V depending on its distance from the obstacle.

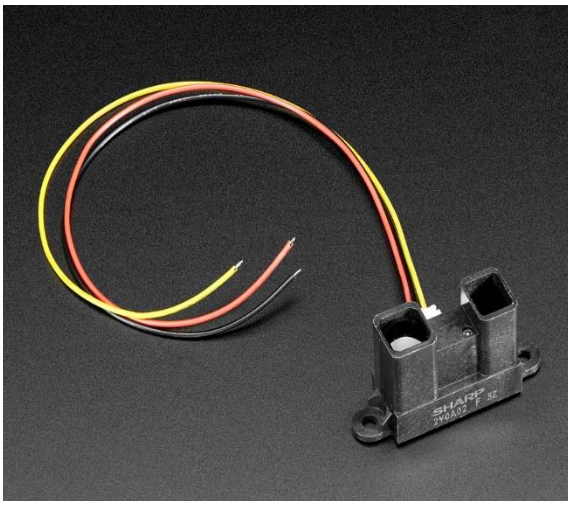

Figure 3: IR Distance Sensor

MicaZ Mote:

Based on Atmel ATmega 128L, a low-power microcontroller, the MicaZ mote supports simultaneous implementations of sensor applications / processing and network / radio communications. Equipped with a 51-pin expansion connector which can connect a variety of sensors and Data Acquisition Boards (MDA-300 CA for this project), the mote can support Analog Inputs, Digital I/O and UART Interfaces through the same.

Figure 4: MicaZ Mote

For implementing the project, we have utilized two such motes one of which (the sensor mote) is connected to the MDA-300 CA Sensor Board through its 51-pin expansion connector.

Principle of Operation:

With the IR Sensor powered through the E5.0 pin which provides an excitation voltage of 5V for the sensor, the latter's output which varies depending on the distance of the obstacle from it is fed into the ADC channel 2 i.e. A2 pin of the ADC.

The MicaZ mote reads the sensor data from the ADC channel through the "IRSensor.readDone(error_t result, uint16_t val)" function every 200 milliseconds which shall be stored in the variable val. Incrementing the value of the variable *iterator* each time the ADC value is read, the values shall be averaged out when the value of the *iterator* variable reaches 10 so as to account for the inaccuracies in output voltages of the sensor. Further, the ADC values are converted into voltages through multiplication by 2.5 and division the variable *volt_denom* which is set at 65000. Glitches, if any, due to float values of the voltages obtained are eliminated by multiplying them by 100.

Additionally, to account for speed and changes associated with it so as to modify the threshold distance over which alarm is given out, a timer "*Timer1*" has been implemented which shall change the speed every 30 seconds by 20. The change in speed, if any, along with the current speed, the calculated sensor output voltage value and the mote ID shall be sent to the Base Station mote and displayed over the terminal screen through Serial Listen.

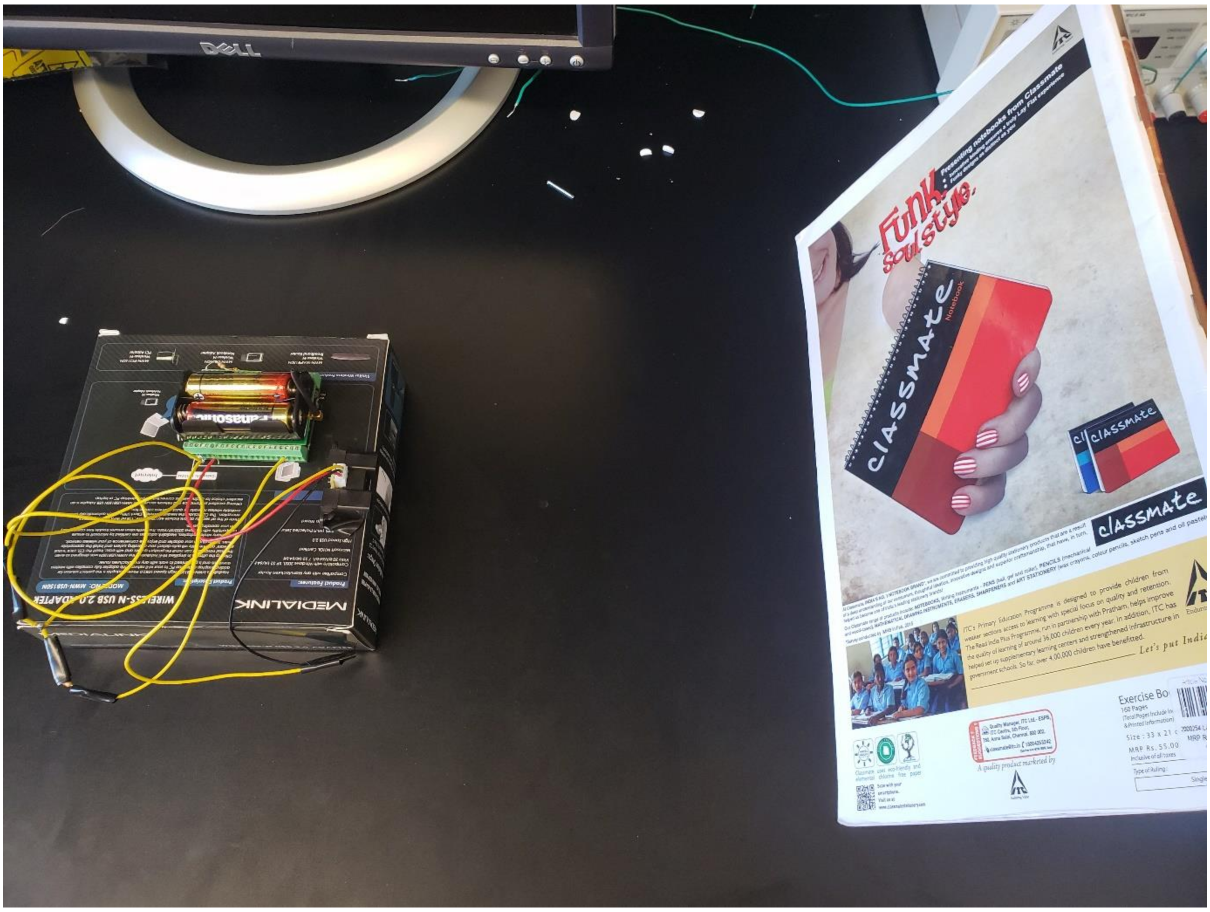

Figure 5: Positioning of the Sensor Mote with IR Sensor

Furthermore, dividing speed by 10, the threshold distance shall be calculated as five times the speed. Having computed sensor output voltage for reference distances during calibration phase of the sensor, the threshold distance is then compared with the reference distance array. Upon threshold distance being less than or equal to reference distance, the sensor output voltage for reference distance is compared with recorded average sensor output voltage. In the event of latter being greater than the former, base station mote shall be duly notified to display alarm over the terminal screen through Serial Listen.

The Base Station, in the meanwhile, receives the data sent by the sensor mote through "*RadioReceive.receive*" function and displays the same, both alarm and voltage messages, over the terminal screen through Serial Listen tool of C.

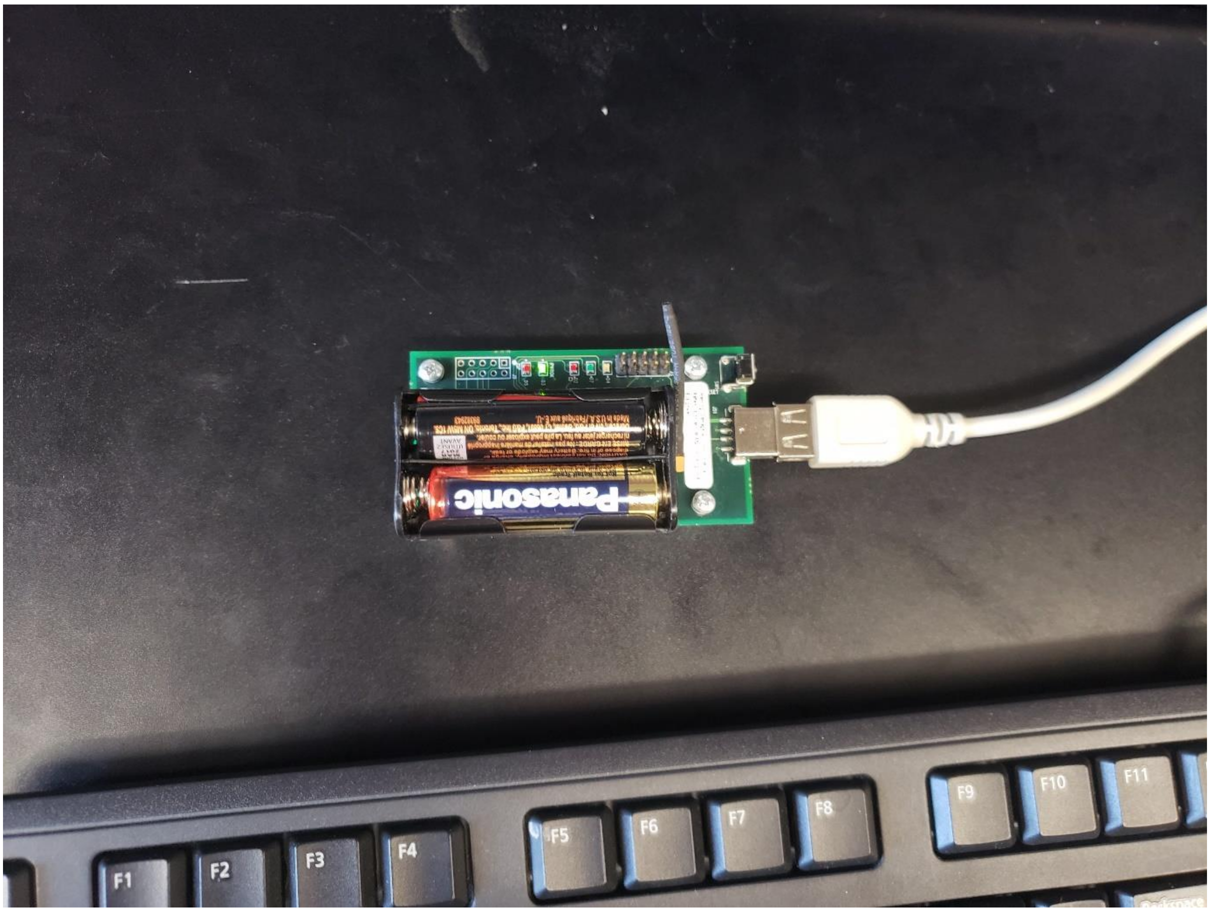

Figure 6: Base Station Mote Connected to the PC through USB for Serial Listen

The Base station program provided in Apps folder of Tiny OS (BaseStationP.nc) has been utilized for the project without any modifications. The Serial Listen tool of C too has been utilized but with modifications relating to displaying data over the Terminal Screen to sate Project requirements.

Algorithm:

1. Calibrate the IR Sensor Offline to create table of reference distances and the voltages displayed for them over the DMM.
2. Programming the Sensor Mote and the Base Station mote.
3. Positioning the Sensor Mote with obstruction at desired distance.
4. Running the Base Station followed by the Serial Listen tool, Turn ON the Sensor Mote.
5. The IR Sensor records the voltage readings in accordance with the moving / stationary obstacle with speed set at 20 into the ADC 2 (A2) channel.
6. The Mote retrieves the readings when the respective Timer fires (Timer 0) and after retrieving 10 readings, averages them to reduce the possible errors in the readings.
7. Voltage values for Sensor Output obtained by multiplying the ADC readings by 2.5 and dividing by 65000.
8. The float Voltage values converted to integers through multiplication with 100.
9. The value of speed divided by 10 and then multiplied by 5 to obtained threshold distance value.
10. The value is compared with the array elements of reference distances considered during offline sensor calibration.
11. The value of voltage for the reference distance which is greater than or equal to the threshold voltage compared with the computed Output voltage of the IR Sensor.
12. If the output voltage value of IR Sensor is greater than the output voltage value for the reference distance at nay given position of the obstacle, alarm displayed over the terminal screen through Base Station and Serial Listen Tool of C.
13. If not, just Sensor ID, the Sensor Output Voltage Value, its corresponding ADC Value, current speed and speed change, if any, displayed.
14. At 30 seconds, Timer 1 fires as a result of which, speed increments by 20, up to maximum of 100, in the aftermath of which, it is reset to 20.

Code Snippet:

Payload:

```
typedef nx_struct ReadSensorMsg {
        nx_uint16_t sensor_id;
        nx_uint16_t value;
        nx_uint16_t speed_change;
        nx_uint16_t new_speed;
} ReadSensorMsg;
```

Timers for reading values from ADC channel and changing speed:

```
event void AMControl.startDone(error_t err) {
                if (err == SUCCESS) {
                        call Timer0.startPeriodic(200);
                        call Timer1.startPeriodic(10000);
                }
                else {
                        call AMControl.start();
                }
        }
```

Distance Detection and Alarm mechanism in Sensor Mote:

```
event void IRSensor.readDone(error_t result, uint16_t val)
{
        iterator++;
        if (iterator < 10) {
        adc_sum = adc_sum + val;
} else {
        adc_sum = adc_sum/10;
        x_voltage = (2.5 * adc_sum)/volt_denom;
```

```
        x_volt_int = x_voltage * 100;
        zz_speed = backup_speed/10;
        threshold_distance = zz_speed * 5;
for (looper=0; looper<5; looper++) {
if (threshold_distance <= dist_arr[looper]) {
        crrnt_val = looper;
        break;
}
}
if (x_volt_int > inv_volt_arr[crrnt_val]) {
        send_stop = 1;
}
if (! busy) {
        ReadSensorMsg* btrpkt = (ReadSensorMsg*) (call Packet.getPayload(&pkt, sizeof
(ReadSensorMsg)));
if (send_stop == 1) {
        btrpkt->sensor_id = 1;
        send_stop = 0;
} else {
        btrpkt->sensor_id = (uint16_t) adc_sum;
}
        btrpkt->value = x_volt_int;
        btrpkt->speed_change = 0;
        if (speed_change ==1) {
        speed_change = 0;
        btrpkt->speed_change = 1;
}
        btrpkt->new_speed = backup_speed;

if (call AMSend.send(AM_BROADCAST_ADDR, &pkt, sizeof(ReadSensorMsg)) ==
SUCCESS) {
        busy = TRUE;
```

```
}
}
if (result == SUCCESS) {
        call Leds.led0Toggle ();
}
        iterator = 0;
}
}
event void Timer0.fired() {
        call IRSensor.read();
}
event void Timer1.fired() {
        call Leds.led2Toggle ();
        speed_change = 1;
        backup_speed = backup_speed + 20;
if (backup_speed > 100) {
        backup_speed = 20;
}
        zz_speed = backup_speed;
}
}
```

Receive Message at Basse Station from Sensor Mote:

```
event message_t *RadioReceive.receive[am_id_t id] (message_t *msg, void *payload, uint8_t
len) {
        return receive (msg, payload, len);
}
```

Display Sensor data and Alarm over terminal Window through Serial Listen Tool

```
if (! packet)
        exit (0);
        data_recvd_1 = ((packet [8] <<8) | (packet [9]));
        data_recvd = ((packet [10] <<8) | (packet [11]));
```

```
        new_speed = ((packet [14] <<8) | (packet [15]));
        if (packet [13] == 1) {
        printf ("\n====== speed changed =======\n");
        printf ("          new speed = %u\n", new_speed);
        printf("============================\n");
}
if (data_recvd_1! = 1) {
        printf("%u-> %u ", data_recvd_1, data_recvd);
} else {
        printf("\n=====================================================
==============");
        printf ("\n                    Brakes Applied                    ");
        printf("\n=====================================================
==============");
}
```

Design Complications:

Varying Output Voltage of IR Sensor

Employing Infrared for detecting obstacles and thereby, giving out alarms, the output of the IR Sensor was observed to fluctuate over same distances for similar dimensions of obstacles. The colour of the latter was also playing a significant part along with a minor part played by fluorescent lights in the vicinity.

Scaling down the IR Sensor Output Voltage

With the maximum voltage of ADC channels of MDA -300 CA being 2.5 V, the output of IR Sensor in response to obstacles higher than 2.5 V were being either displayed as zero or as random numbers.

Floating point Values of voltages calculated from ADC Values:

With the read function for MDA-300 CA with MicaZ taking a *uint* variable as parameter, the float values of voltages calculated from ADC Values were erroneously displayed.

Implemented Solutions:

Offline Calibration of IR Sensor:

With the objective of mitigating the variable values of IR Sensor in view, offline calibration of the same was undertaken by powering the sensor through the MDA and taking voltage readings for obstacles at varying distances. The readings, since they were varying, were taken for obstacles of different colours, predominantly white and black under varying light conditions. The best recorded values, as shown in table below, were then selected as reference voltages for these distances and employed in distance detection and alarm logic.

Distance from obstacle	Voltage Reading (V)
No Obstacle	1.033
2	0.96
4	1.04
6	1.14
8	1.127
10	2.04
15	2.23
20	2.04
25	1.88
30	1.63
40	1.3
50	1.1
60	0.99
70	1.02
80	1.01
90	1.01
100	1.02

Table 1: Offline Sensor Calibration

Implementation of Voltage Divider

With the peak output voltage of the IR Sensor exceeding the maximum voltage of the ADC Channels of MDA-300 CA, a voltage Divider was implemented at the output of the IR Sensor to scale down the very voltage below 2.5 V. Selecting values of R1 and R2 as 2K ohms and 17K ohms, the voltage divider successfully scaled down the output voltage of the IR Sensor to around 2.25 to 2.3 V.

Multiplying Voltages computed by 100

Averaging the ADC values to further mitigate effects of variable output voltages of the IR Sensor, the erroneous values displayed due to incompatible datatypes of receive function and voltages computed from ADC values was successfully resolved by multiplying the calculated voltages by 100 i.e. output voltages obtained over the terminal were the output voltages of the sensor multiplied by 100.

Execution:

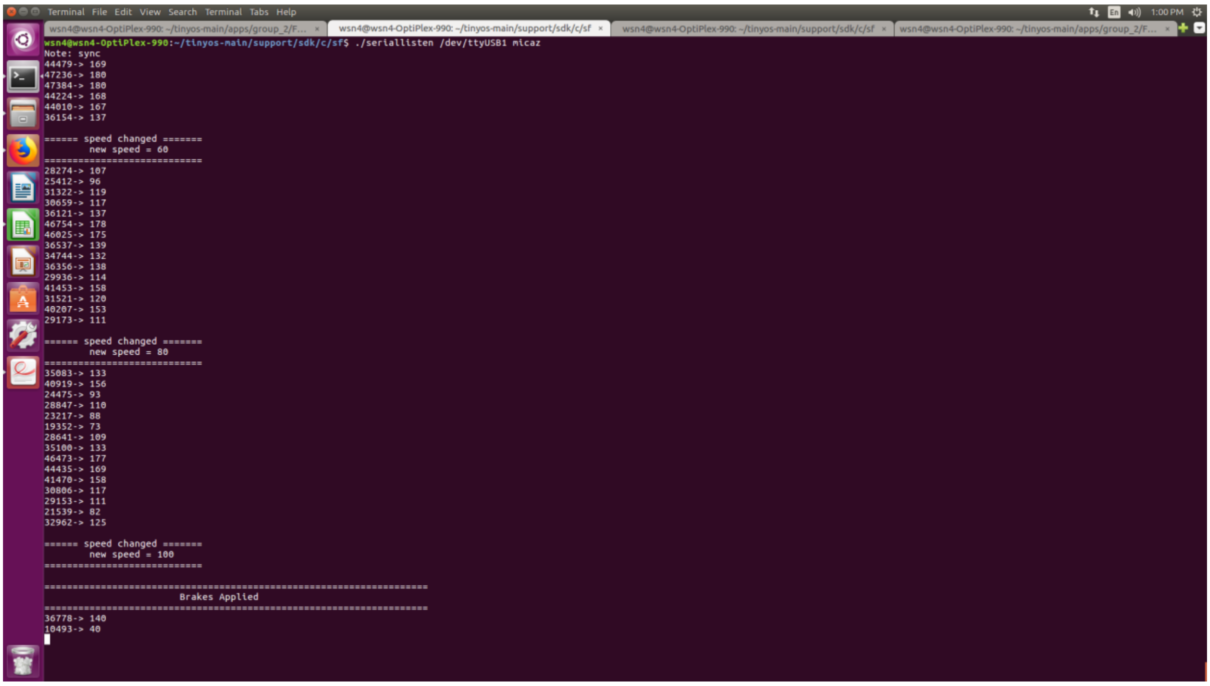

Figure 7: Terminal Output with Serial Listen

As is evident from the figure above, starting up the both motes and running Serial Listen, the output voltages of the IR Sensor are obtained along with the corresponding ADC values as the obstacles are moved within sensor range and angle. Also displayed the event of change in speed along with the new speed.

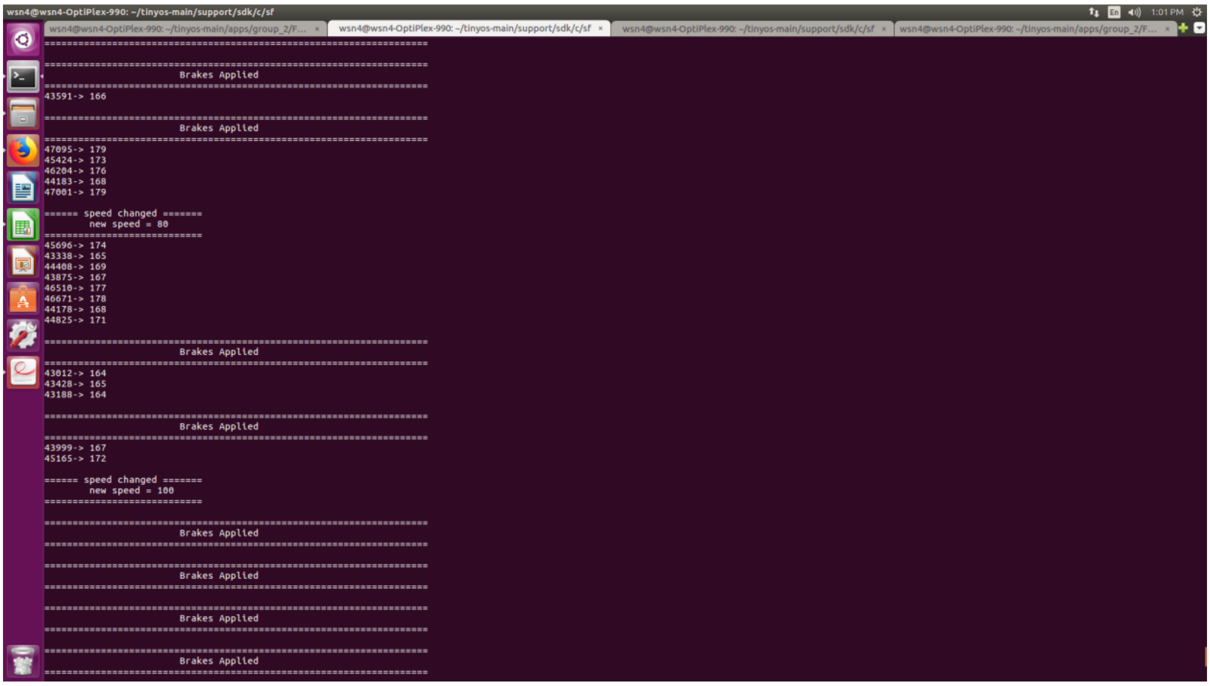

Figure 8: Terminal Output with Alarm Display

Allowing the program to run continuously, the speed goes on incrementing with 20 till hundred in the aftermath of which it is reset. Also, with the threshold distance increasing at high speeds, alarms are given off continuously at greater distances than that for lower speeds until the obstacle is moved further away from the sensor or out of its angle.

Future Scope:

With the ADC Channel of the MDA-300 CA occasionally providing random values for Sensor input voltages to it, in spite of several measures attempted to eliminate the same, and with phenomenon repeating for other ADC Channels of the Sensor Board, noted through observations, resulting in erroneous computed voltages of the Sensor, a direction for future work pertaining to the project could be pinpointing the glitch, as the same is not observed when the channels are provided with input voltages through an external power supply, and eliminating the same.

Another direction of future work could be to send the values of speed from the terminal through Serial Write Tool to the Sensor Mote for it to make its computations and provide alarms for distances accordingly.

Finally, while, in the current status, the mote transmits the sensor output voltages continuously, a large majority of which are plain voltages with little to observe, an energy efficient implementation of the same could be implemented in which only occurrence of alarm events is transmitted back to the base station for terminal display.

Conclusion:

The Vehicle Distance Detection and alarm has thus, been implemented. With the IR Sensor sensing and providing proportional output voltages to be stored in ADC Channel (A2) of MDA-300 CA Data Acquisition Board, the same are read by the MicaZ mote to compute the sensor output voltages. With the speed provided through a variable and incremented through firing of a timer, the new speed and the change in speed, if any, is sent back to the base station mote along with Sensor output Voltages for terminal display through Serial Listen. Utilizing the speed provided to compute the distance threshold, the same in combination with output voltages obtained for reference distances and computed sensor output voltage values are utilized to detect alarm events, which too are transmitted to the base station for terminal display, depending upon the distance threshold and computed sensor output voltages i.e. an alarm shall be displayed at greater distances for higher speeds than at lower speeds.