<SECURITY SYSTEM FOR VEHICLES FROM ANY THEFT >

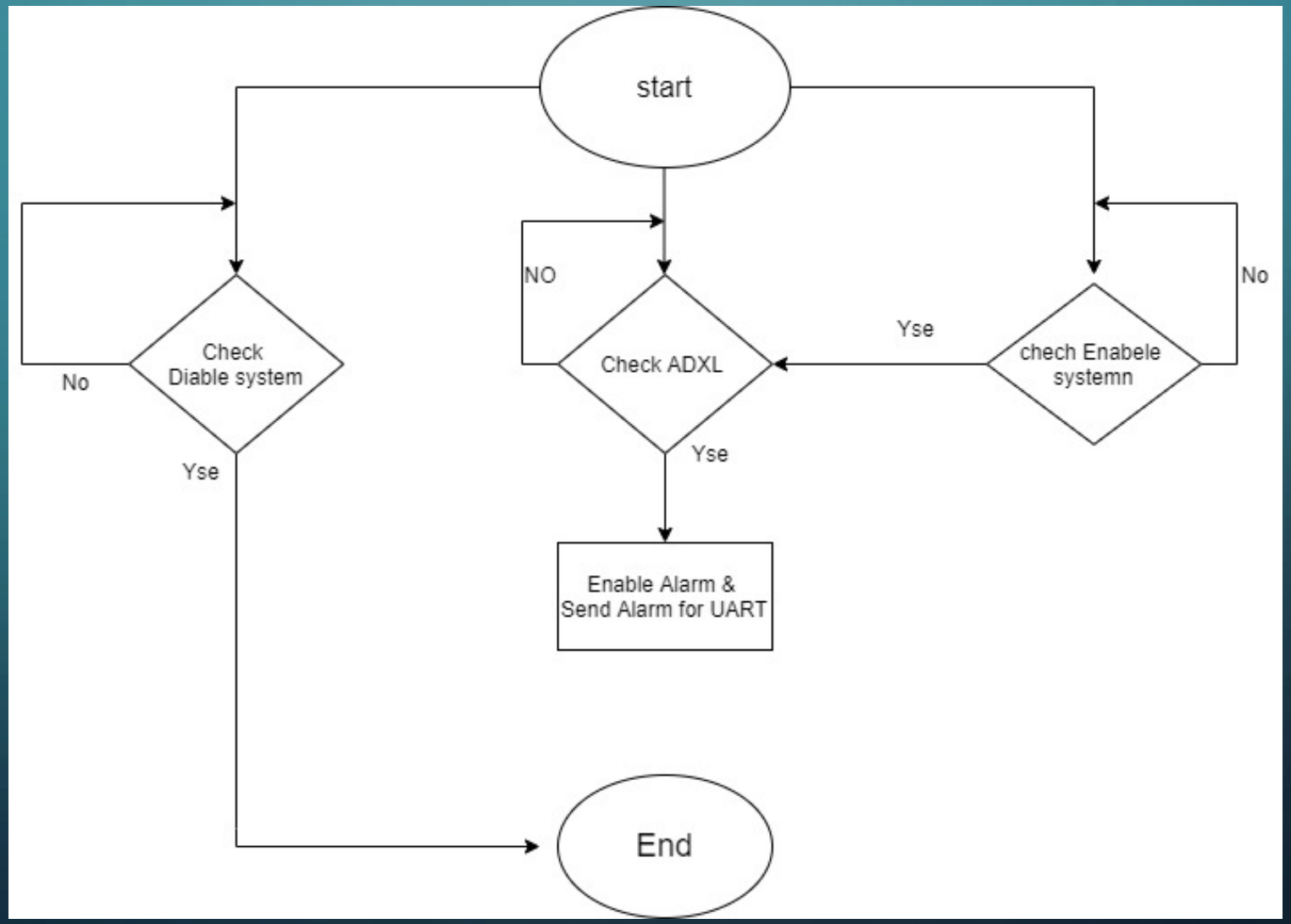
THE PROBLEM

What is the problem?	• Car security measures are very important because **cars get very often stolen and broken into and more often than not**
Who has this problem?	• Cars' owners , cars manufactures
Why should this problem be solved?	• Consumer Protection • increase car sales
How will I know this problem has been solved?	• When we have a real time alarming

REQUIREMENTS :

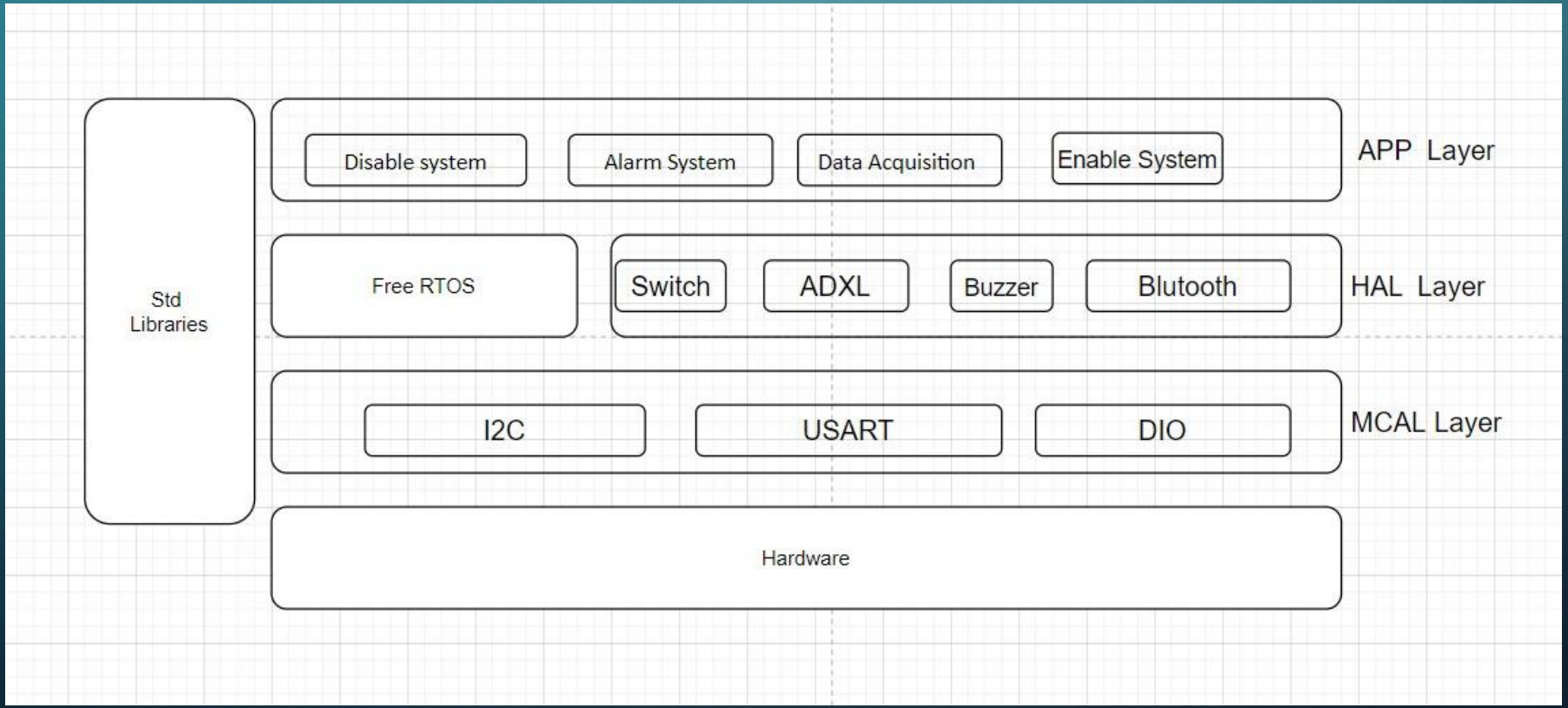
- The user can enable and disable the software using switch/ UART .
- the system can detect if the is a theft or any intrusion .
- The system sends to the user if there is any attack .
- The system receives a response from the user to make actions.
- The system enable alarm system , if received a negative response from the user .

FLOW CHART FOR THE WORKABLE SOLUTION

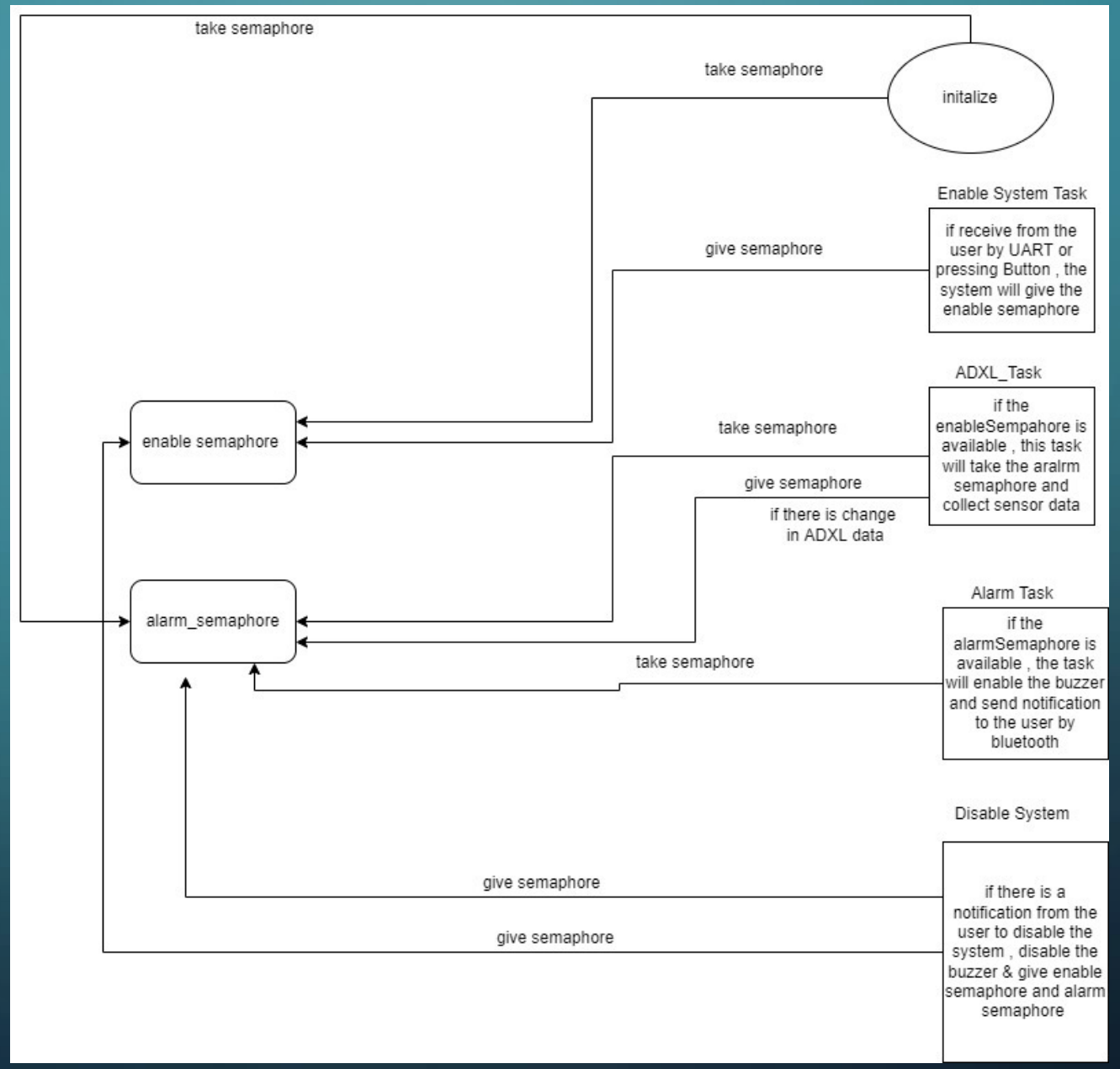


SYSTEM DESIGN :

- Static design



- Dynamic design



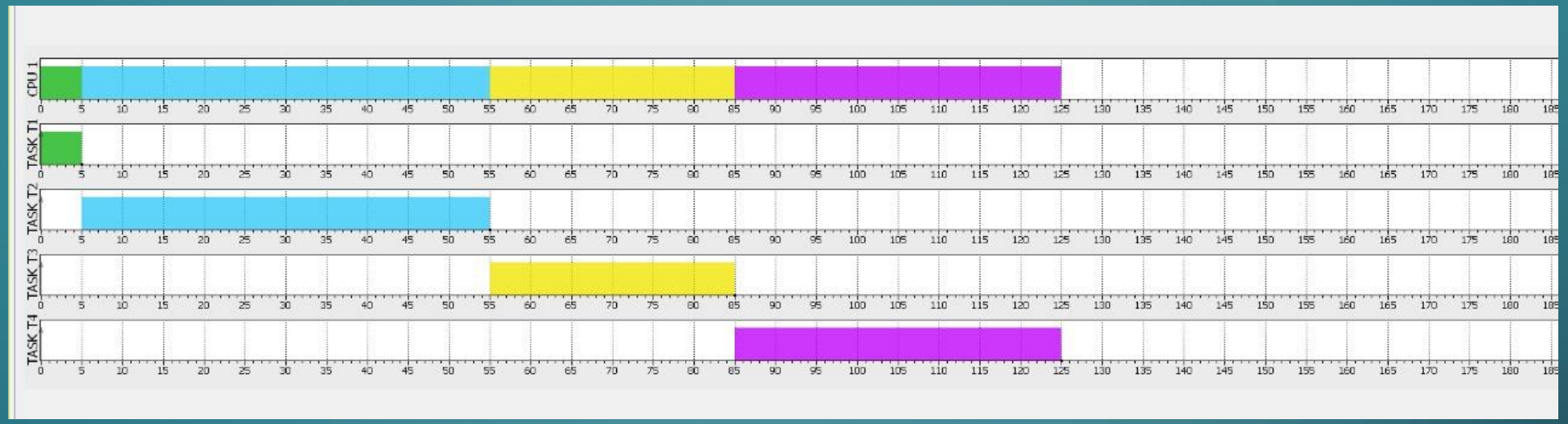
TASKS :

- the system need 4 tasks in the system:

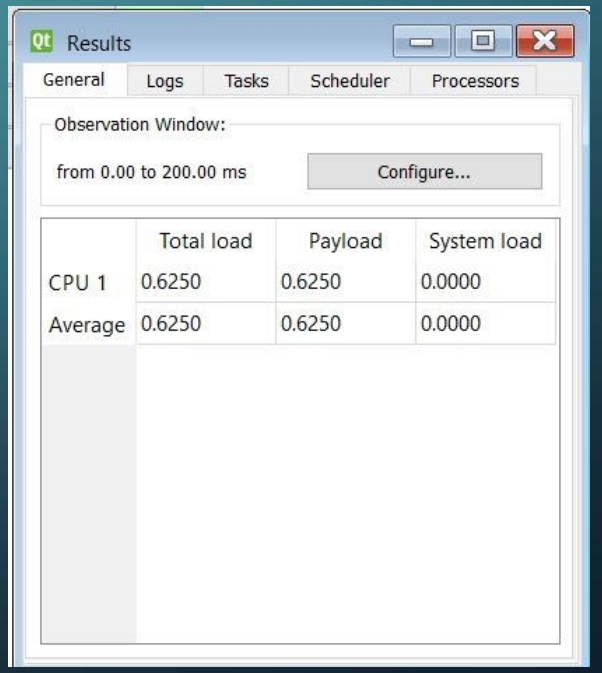
the system need 4 tasks in the system:

Task name	***priority***	***periodicity***	***execution time***
APP Enable Software Task	4	400 msec	**5 msec**
APP Data Acquisition Task	3	1 sec	**50 msec**
APP Alarm Task	2	1 sec	**30 msec**
APP Display Task	1	1sec	**40 msec**

TASKS OFFLINE SIMULATION :



- System Tick = 200 msec
- CPU load = 62%



THE SYSTEM CONSISTS OF :

- Data array (for save the system data in one container)

the array consists of

- three integers for save acceleration sensor data
  - ➔ Integer X: for the first dimensions.
  - ➔ Integer Y: for the second dimensions.
  - ➔ Integer Z: for the third dimensions.

- **Semaphores :**

we need 2 “Binary” semaphores in the system:

➔Enable Software Semaphore (for block or unblock all system).

➔Alarm Semaphore (for block or unblock the Alarm subsystem).

TESTING THE PROTOTYPE

- Task 1: Enable Software This task controls in all the software for enable the system or not, that is make the system is more flexible.

```
110 void APP_Enable_System(void *pv)
111 {
112     u8 ButtonState;
113     u8 Loc_SemStartState;
114
115     while(1)
116     {       Loc_SemStartState=xSemaphoreTake(EnabelSem,10);
117     MDIO_Error_State_GetPinValue(PIN1,MDIO_PORTC,&ButtonState);
118
119     if(ButtonState==0)
120     {
121         xSemaphoreGive(EnabelSem);
122         StartFlag=1;
123         StopFlag=0;
124     }
125     else if(StartFlag==1)
126     {
127         xSemaphoreGive(EnabelSem);
128         StopFlag=0;
129     }
130     vTaskDelay(400);
131     }
132 }
133
```

- Task 2: **Data Acquisition** This task work to collect the from sensors , The data consists of acceleration data. *This task stores the required data in one container.*

```
void APP_ADXL(void *Pv){
    u8 Loc_SemADXLState;
    float Frist_Read_Acc_Data[3]={0};
    float Second_Read_Acc_Data[3]={0};

    ADXL_Acc(Frist_Read_Acc_Data);

    while(1){
        Loc_SemADXLState=xSemaphoreTake(EnabelSem,10);
        if(Loc_SemADXLState==pdPASS){
            xSemaphoreTake(FlagAlarmSem,10);
            //Check Data from Sensor
            ADXL_Acc(Second_Read_Acc_Data);
            if(Frist_Read_Acc_Data[0]!=Second_Read_Acc_Data[0]
                || Frist_Read_Acc_Data[1]!=Second_Read_Acc_Data[1]
                || Frist_Read_Acc_Data[2]!=Second_Read_Acc_Data[2])
            {

                xSemaphoreGive(FlagAlarmSem);

            }
            else{
                /*Do Nothing*/
            }

        }
        else{
            /*Do Nothing*/
        }
        vTaskDelay(1000);
    }


}
```

- Task 3: **Alarm system** This task for enabling the alarm system

```
170
171 void APP_Alarm(void *Pv){
172     u8 Loc_SemUARTSendState;
173
174     while(1){
175         Loc_SemUARTSendState=xSemaphoreTake(FlagAlarmSem,10);
176         if(Loc_SemUARTSendState==pdPASS){
177
178             /*Enable Buzzer*/
179             MDIO_Error_State_SetPinValue(PIN1,MDIO_PORTB,PIN_HIGH);
180
181             /*Send Alarm on UART*/
182             MUSART_VidSendString("Warning..!!!");
183
184         }
185         else{
186             /*Do Nothing*/
187         }
188         vTaskDelay(1000);
189
190     }
191
192 }
193
```

- Task 4: **Disable system** This task is virtual task, we use it for monitoring the system data.

```
195 void APP_Disable_System(void *Pv){
196
197     while(1){
198
199         if((StopFlag==1)&&(StartFlag==1))
200         {
201             /*Disable Buzzer*/
202             MDIO_Error_State_SetPinValue(PIN1,MDIO_PORTB,PIN_LOW);
203
204             xSemaphoreGive(EnabelSem);
205             xSemaphoreGive(FlagAlarmSem);
206
207             StartFlag=0;
208
209         }
210         else{
211             /*Do Nothing*/
212         }
213         vTaskDelay(1000);
214
215     }
216
217 }
218
```

THE PROTOTYPE COMPONENTS

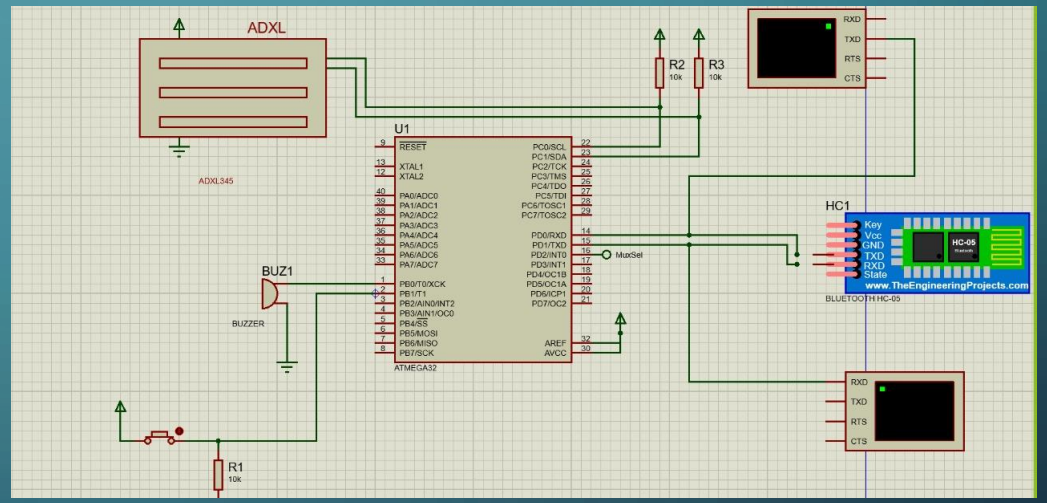
1-ATMEGA32

2-Switch

3-Buzzer

4-Bluetooth

5- Virtual ADXL



Thank you

Mukhtar Mohammed

Mohammed hamada

Mohammed Abdallah

Marwa Shokry

Zeinab Muhammad