**Interfacing PIR Sensor with PIC Microcontroller(Motion Detection)- Smart Home**

## Aim:

PIR (Passive Infrared) or Motion Sensor is used to detect the motion of moving human body or objects. Whenever someone comes in the range of PIR sensor, it gives High at its output pin. **we are simple going to interface PIR with PIC Microcontroller**PIC16F877A. In this circuit if some moving objects comes in the range of PIR sensor, the buzzer will start beeping.

## **Material Required**

* PicKit 3
* PIR Sensor.
* PIC16F877A IC
* 40 - Pin IC holder
* Perf board
* 20 MHz Crystal OSC
* Female and Male Bergstick pins
* 33pf Capacitor - 2Nos, 100uf and 10uf cap.
* 680 ohm, 10K and 560ohm Resistor
* LED of any color
* 1 Soldering kit
* IC 7805
* 12V Adapter
* Buzzer
* Connecting wires
* Breadboard

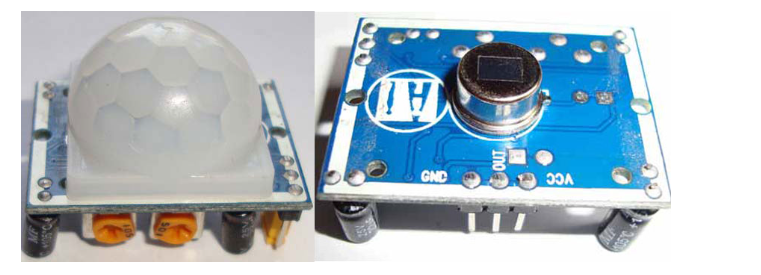
### Team Members:

* Kalyan A – CB.EN.U4ELC22004
* Sai Pavan – CB.EN.U4ELC22011
* K Praveen – CB.EN.U4ELC22016

### Batch No.: 14

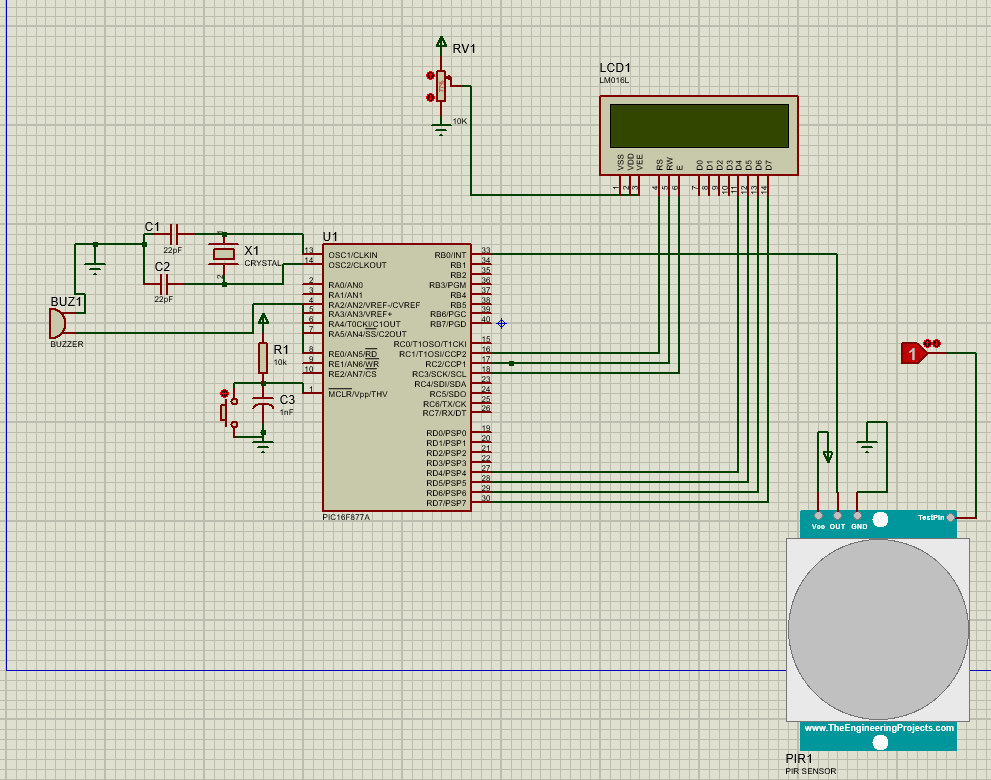
## **PIR Sensor:**

* PIR sensor is inexpensive, low-power and easy to use Motion Detections Sesnor. PIR sensor only receives infrared rays, not emits that’s why it’s called passive. PIR sense any change in heat, and if there is a change it gives HIGH at OUTPUT. PIR Sensor also referred as Pyroelectric or IR motion sensor.



* Every object emits some amount of infrared when heated, similar to that human body emits IR due to body heat. Infrared created by every object because of the friction between air and object. The main component of PIR sensor is Pyroelectric sensor. Along with this, [BISS0001 ("Micro Power PIR Motion Detector IC"](http://www.ladyada.net/media/sensors/BISS0001.pdf)), some resistors, capacitors and other components used to build PIR sensor. BISS0001 IC take the input from sensor and does processing to make the output pin HIGH or LOW accordingly.
* [Learn more about PIR sensor here](https://circuitdigest.com/electronic-circuits/pir-sensor-based-motion-detector-sensor-circuit). You can also adjust distance sensitivity and time duration for which the output pin will be high once motion is detected. It has two potentiometer knobs to adjust these two parameters.

## CIRCUIT DIAGRAM:



## Port Connections:

### 1. LCD Control Pins:

• RS (Register Select): Connected to RC1

• RW (Read/Write): Connected to RC0

• E (Enable): Connected to RC2

### 2. LCD Data Pins:

• Data pins (D0-D7): Connected to PORTD

### 3. PIR Sensor:

• Output: Connected to RB0 (input pin)

### 4. Buzzer:

• Control: Connected to RE0 (output pin)

5. Matrix Keyboard:

* RB5=0, RB6=1, RB7=1; RB1=0, RB2=0;

Input from user

## Main Loop Structure:

**Polling for Switches:**The main loop continually checks for switches RB1 and RB2 to be pressed. When either switch is pressed, it enters a nested while loop checking for PIR sensor activity (RB0).**PIR Sensor Handling:**If motion is detected (RB0 == 1), the LCD displays "ALERT! INTRUDED" and the buzzer is turned on.If no motion is detected, the LCD displays "SAFE" and the buzzer is turned off.

### Source Code:

#include <xc.h> // Header file for the XC8 compiler

int flag = 0;

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Declarations

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

void delay\_ms(int ms);

void Data(int Value);

void Cmd(int Value);

void Send2Lcd(const char Adr, const char \*Lcd);

void initLCD(void);

void adc\_interrupt(void);

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Main Program

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

void \_\_interrupt() adc\_interrupt() {

if (ADIF == 1) {

flag = 1;

ADIF = 0;

}

}

void main(void) {

nRBPU=0;

PCFG3=0;

PCFG2=1;

PCFG1=1;

PCFG0=0;

TRISB=0b00011111;

// Port initialization

TRISC = 0x00; // PORTC as output (used for LCD control)

TRISD = 0x00; // PORTD as output (used for LCD data)

TRISE0 = 0; // RE0 as output (connected to Buzzer)

PORTC = 0x00; // Clear PORTC

PORTE = 0x00; // Clear PORTE

// Initialize LCD

initLCD();

while (1) {

Send2Lcd(0x80, "Press SW1 or SW4"); // LCD Data Outing Function

RB5=0;

RB6=1;

RB7=1;

if (RB1 == 0) {

while (RB1 == 0) {

while (1) {

if (RB0 == 1) { // If PIR sensor detects motion

delay\_ms(15); // Minimum Delay To Power On LCD Module To Recieve Mode

Cmd(0X30); delay\_ms(5); // LCD Specification Commands

Cmd(0X30); delay\_ms(1); // LCD Specification Commands

Cmd(0X30); delay\_ms(2); // LCD Specification Commands

Cmd(0X38); // LCD Double Line Display Command

Cmd(0X06); // LCD Auto Increment Location Address Command

Cmd(0X01); // LCD Display Clear Command

Cmd(0X0C); // LCD Display ON Command

Send2Lcd(0x80, "ALERT! INTRUDED"); // LCD Data Outing Function

RE0 = 1; // Turn on buzzer

while (1) {

GO = 1;

Cmd(0xC0);

}

} else {

delay\_ms(15); // Minimum Delay To Power On LCD Module To Recieve Mode

Cmd(0X30); delay\_ms(5); // LCD Specification Commands

Cmd(0X30); delay\_ms(1); // LCD Specification Commands

Cmd(0X30); delay\_ms(2); // LCD Specification Commands

Cmd(0X38); // LCD Double Line Display Command

Cmd(0X06); // LCD Auto Increment Location Address Command

Cmd(0X01); // LCD Display Clear Command

Cmd(0X0C); // LCD Display ON Command

Send2Lcd(0x80, "SAFE"); // LCD Data Outing Function

delay\_ms(3);

RE0 = 0; // Turn off buzzer

}

delay\_ms(100); // Delay to avoid flickering

}

}

}

if (RB2 == 0) {

while (RB2 == 0) {

while (1) {

if (RB0 == 1) { // If PIR sensor detects motion

delay\_ms(15); // Minimum Delay To Power On LCD Module To Recieve Mode

Cmd(0X30); delay\_ms(5); // LCD Specification Commands

Cmd(0X30); delay\_ms(1); // LCD Specification Commands

Cmd(0X30); delay\_ms(2); // LCD Specification Commands

Cmd(0X38); // LCD Double Line Display Command

Cmd(0X06); // LCD Auto Increment Location Address Command

Cmd(0X01); // LCD Display Clear Command

Cmd(0X0C); // LCD Display ON Command

Send2Lcd(0x80, "ALERT! INTRUDED"); // LCD Data Outing Function

RE0 = 1; // Turn on buzzer

} else {

delay\_ms(15); // Minimum Delay To Power On LCD Module To Recieve Mode

Cmd(0X30); delay\_ms(5); // LCD Specification Commands

Cmd(0X30); delay\_ms(1); // LCD Specification Commands

Cmd(0X30); delay\_ms(2); // LCD Specification Commands

Cmd(0X38); // LCD Double Line Display Command

Cmd(0X06); // LCD Auto Increment Location Address Command

Cmd(0X01); // LCD Display Clear Command

Cmd(0X0C); // LCD Display ON Command

Send2Lcd(0x80, "SAFE"); // LCD Data Outing Function

delay\_ms(3);

RE0 = 0; // Turn off buzzer

}

delay\_ms(50); // Delay to avoid flickering

}

}

}

}

}

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function : initLCD

\* Description : Function to initialize the LCD

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

void initLCD(void) {

delay\_ms(15); // Minimum delay to power on LCD module to receive mode

Cmd(0x30); // LCD Specification Commands

delay\_ms(5);

Cmd(0x30); // LCD Specification Commands

delay\_ms(1);

Cmd(0x30); // LCD Specification Commands

delay\_ms(2);

Cmd(0x38); // LCD Double Line Display Command

Cmd(0x06); // LCD Auto Increment Location Address Command

Cmd(0x01); // LCD Display Clear Command

Cmd(0x0C); // LCD Display ON Command

}

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function : Cmd

\* Description : Function to send a command to LCD

\* Parameters : Value, contains the command to be sent

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

void Cmd(int Value) {

PORTD = (unsigned char)Value;

RC1 = 0; // RC1=0(RS=0) [Command Register Selection]

RC0 = 0; // RC0=0(R/W=0) [Write Process]

RC2 = 1; // RC2=1(Enable=1) [Enable Line ON]

delay\_ms(4); // Minimum delay for hold on data

RC2 = 0; // RC2=0(Enable=0) [Enable Line OFF]

}

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function : Data

\* Description : Function to send a data to LCD

\* Parameters : Value, contains the data to be sent

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

void Data(int Value) {

PORTD = (unsigned char)Value;

RC1 = 1; // RC1=1(RS=1) [Data Register Selection]

RC0 = 0; // RC0=0(R/W=0) [Write Process]

RC2 = 1; // RC2=1(Enable=1) [Enable Line ON]

delay\_ms(4); // Minimum delay for hold on data

RC2 = 0; // RC2=0(Enable=0) [Enable Line OFF]

}

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function : Send2Lcd

\* Description : Function to send a string of data to LCD

\* Parameters : adr, contains the address of a string

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

void Send2Lcd(const char Adr, const char \*Lcd) {

Cmd(Adr);

while (\*Lcd != '\0') {

Data(\*Lcd);

Lcd++;

}

}

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function : delay\_ms

\* Description : Function for delay

\* Parameters : ms, contains milliseconds

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

void delay\_ms(int ms) {

int i, count;

for (i = 1; i <= ms; i++) {

count = 1000;

while (count != 0) {

count--;

}

}

}