**FLOOD MONITORING ALARM SYSTEM**

Joshua S. Rebuta

Mark Nelson A. Bentulan

Emman Ace G. Menion

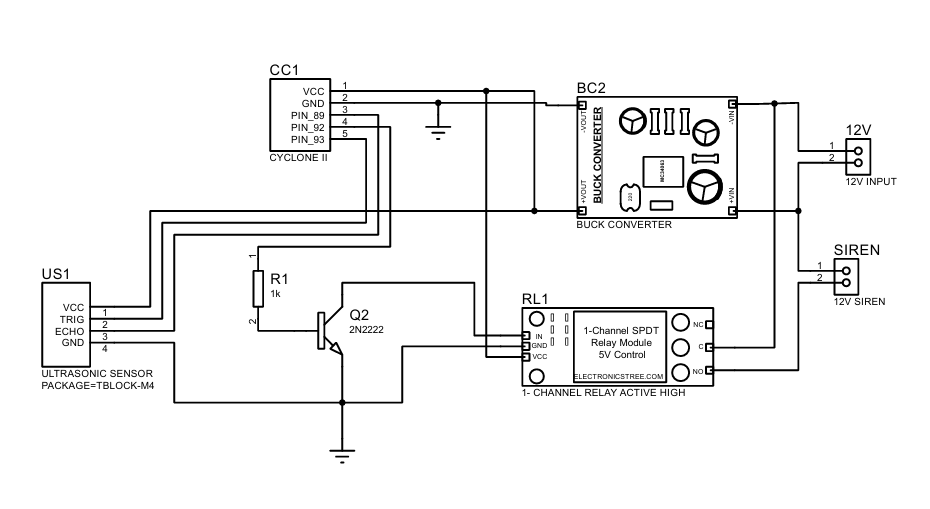
Welbert C. Lumapguid



The device is aim to monitor the water level of rivers or other bodies of water. If the water rises to a critical level, the device will detect it and trigger an alarm through a siren. Motor horn is used as a siren in the device. An Ultra Sonic Sensor (HC-SR04T) is used to detect the water level.

The device is plan to be installed where the sensor is facing the water in a downward position. One example is installing the device under a bridge of a river where overflow that leads to flooding may happen.

**System’s Schematic Diagram**



**Electrical components used:**

* Ultra Sonic Sensor (HC-SR04T)
* 12V motor horn
* 2n2222 transistor
* FPGA Cyclone 2 Altera (EP2C5T144C8N)
* Buck converter (DC to DC)
* 5V relay module (1 channel)

**SYSTEM’S CODE (FPGA Cyclone 2)**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

entity sample is

Port (

clk : in STD\_LOGIC; -- 50MHz clock

trig : out STD\_LOGIC; -- HC-SR04 Trigger

echo : in STD\_LOGIC; -- HC-SR04 Echo

relay : out STD\_LOGIC -- Relay output (PIN\_92)

);

end sample;

architecture Behavioral of sample is

-- Constants for 50MHz operation

constant CLK\_FREQ : integer := 50\_000\_000;

constant SOUND\_SPEED : integer := 34300; -- cm/s

constant DIST\_ON : integer := 35;

constant DIST\_OFF : integer := 36;

-- Timing constants

constant TRIG\_CYCLES : integer := 500; -- 10µs trigger

constant MEASURE\_DELAY : integer := 1\_500\_000; -- 30ms between measurements

constant ECHO\_TIMEOUT : integer := 750\_000; -- 15ms max echo timeout

constant RELAY\_TIME : integer := 250\_000\_000; -- 5s hold time

-- Debouncing

constant NUM\_SAMPLES : integer := 5; -- Consistent readings required

constant SAMPLE\_DEV : integer := 1; -- Max deviation between samples (cm)

-- State machine

type state\_type is (IDLE, TRIGGER, WAIT\_ECHO, MEASURE, VERIFY);

signal state : state\_type := IDLE;

-- Synchronized echo input

signal echo\_sync : std\_logic\_vector(2 downto 0) := "000";

-- Measurement system

signal counter : integer range 0 to MEASURE\_DELAY := 0;

signal echo\_timer : integer range 0 to ECHO\_TIMEOUT := 0;

signal distance\_cm : integer range 0 to 500 := 0;

-- Precision detection system

type sample\_array is array (0 to NUM\_SAMPLES-1) of integer range 0 to 500;

signal samples : sample\_array := (others => 0);

signal sample\_index : integer range 0 to NUM\_SAMPLES-1 := 0;

signal relay\_reg : std\_logic := '0';

signal hold\_timer : integer range 0 to RELAY\_TIME := 0;

signal object\_detected : std\_logic := '0';

signal power\_on\_reset : std\_logic := '1';

signal reset\_counter : integer range 0 to 1\_000\_000 := 0; -- 20ms reset at 50MHz

begin

process(clk)

begin

if rising\_edge(clk) then

if reset\_counter < 1\_000\_000 then

reset\_counter <= reset\_counter + 1;

power\_on\_reset <= '1';

else

power\_on\_reset <= '0';

end if;

end if;

end process;

process(clk)

begin

if rising\_edge(clk) then

echo\_sync <= echo\_sync(1 downto 0) & echo;

end if;

end process;

process(clk)

variable sample\_avg : integer range 0 to 500;

variable consistent : boolean;

begin

if rising\_edge(clk) then

if power\_on\_reset = '1' then

-- Initialize all signals (like JTAG mode)

relay\_reg <= '0';

object\_detected <= '0';

hold\_timer <= 0;

state <= IDLE;

counter <= 0;

echo\_timer <= 0;

samples <= (others => 0);

else

trig <= '0'; -- Default trigger off

case state is

when IDLE =>

if object\_detected = '1' then

-- Hold relay ON for RELAY\_TIME

if hold\_timer < RELAY\_TIME then

hold\_timer <= hold\_timer + 1;

else

hold\_timer <= 0;

object\_detected <= '0';

relay\_reg <= '0';

end if;

elsif counter < MEASURE\_DELAY then

counter <= counter + 1;

else

counter <= 0;

state <= TRIGGER;

end if;

when TRIGGER =>

trig <= '1';

if counter < TRIG\_CYCLES then

counter <= counter + 1;

else

counter <= 0;

state <= WAIT\_ECHO;

end if;

when WAIT\_ECHO =>

if echo\_sync(2) = '1' then

echo\_timer <= 0;

state <= MEASURE;

elsif counter < ECHO\_TIMEOUT then

counter <= counter + 1;

else

state <= IDLE; -- Timeout

end if;

when MEASURE =>

if echo\_sync(2) = '1' then

if echo\_timer < ECHO\_TIMEOUT then

echo\_timer <= echo\_timer + 1;

else

state <= IDLE; -- Timeout

end if;

else

distance\_cm <= (echo\_timer \* SOUND\_SPEED + CLK\_FREQ) /(2\*CLK\_FREQ);

state <= VERIFY;

end if;

when VERIFY =>

samples(sample\_index) <= distance\_cm;

sample\_index <= (sample\_index + 1) mod NUM\_SAMPLES;

consistent := true;

sample\_avg := (samples(0) + samples(1) + samples(2) +

samples(3) + samples(4)) / NUM\_SAMPLES;

for i in 0 to NUM\_SAMPLES-1 loop

if abs(samples(i) - sample\_avg) > SAMPLE\_DEV then

consistent := false;

end if;

end loop;

if consistent then

if sample\_avg <= DIST\_ON and object\_detected = '0' then

relay\_reg <= '1';

object\_detected <= '1';

hold\_timer <= 0;

elsif sample\_avg >= DIST\_OFF and object\_detected = '1' then

relay\_reg <= '0';

object\_detected <= '0';

end if;

end if;

state <= IDLE;

when others =>

state <= IDLE;

end case;

end if;

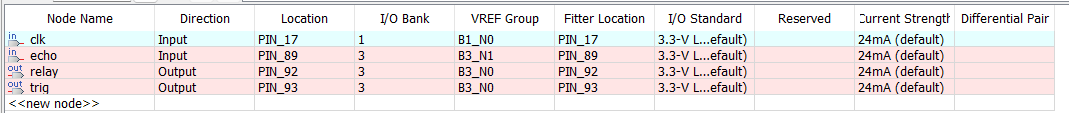
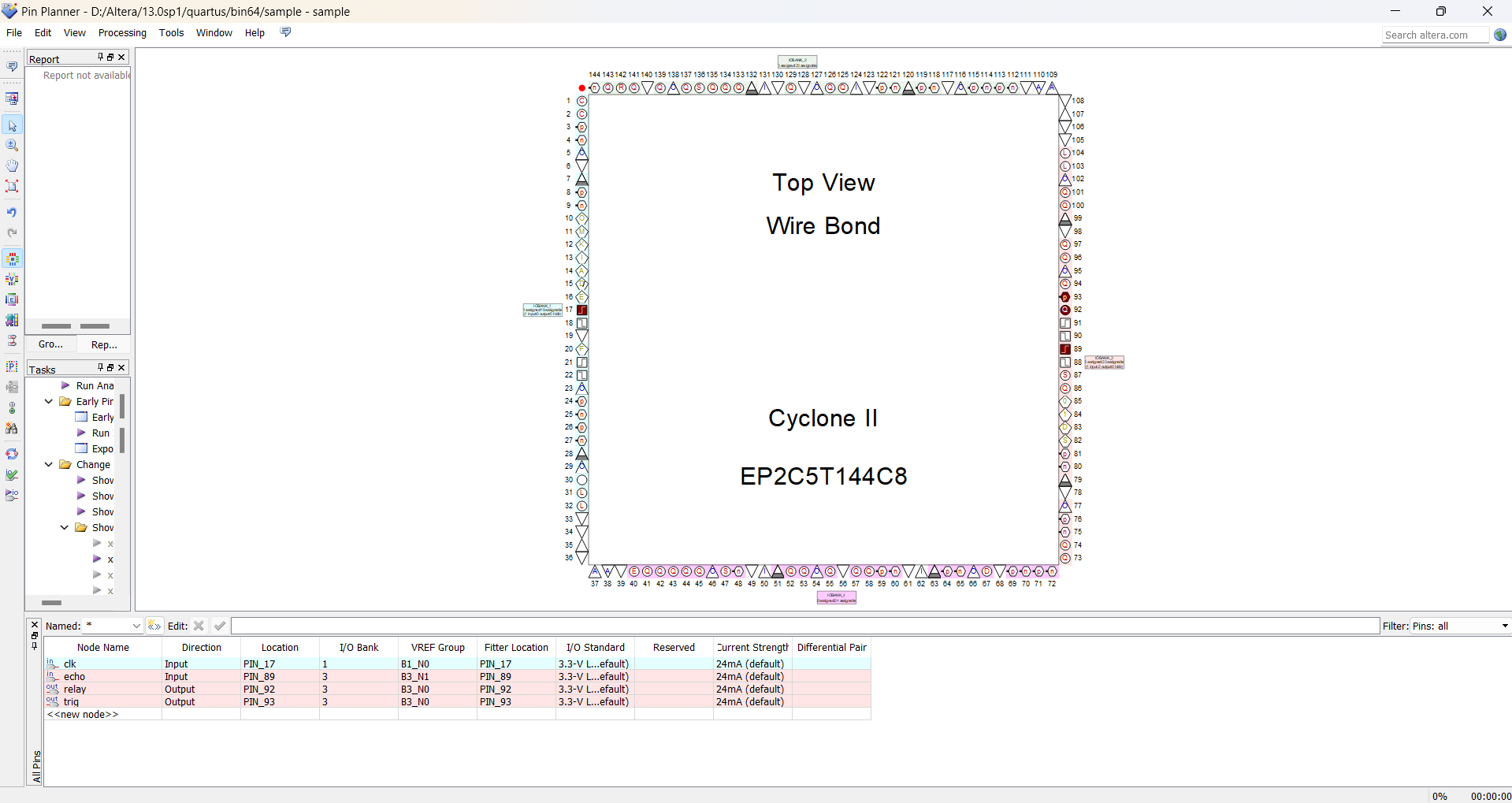
relay <= relay\_reg;

end if;

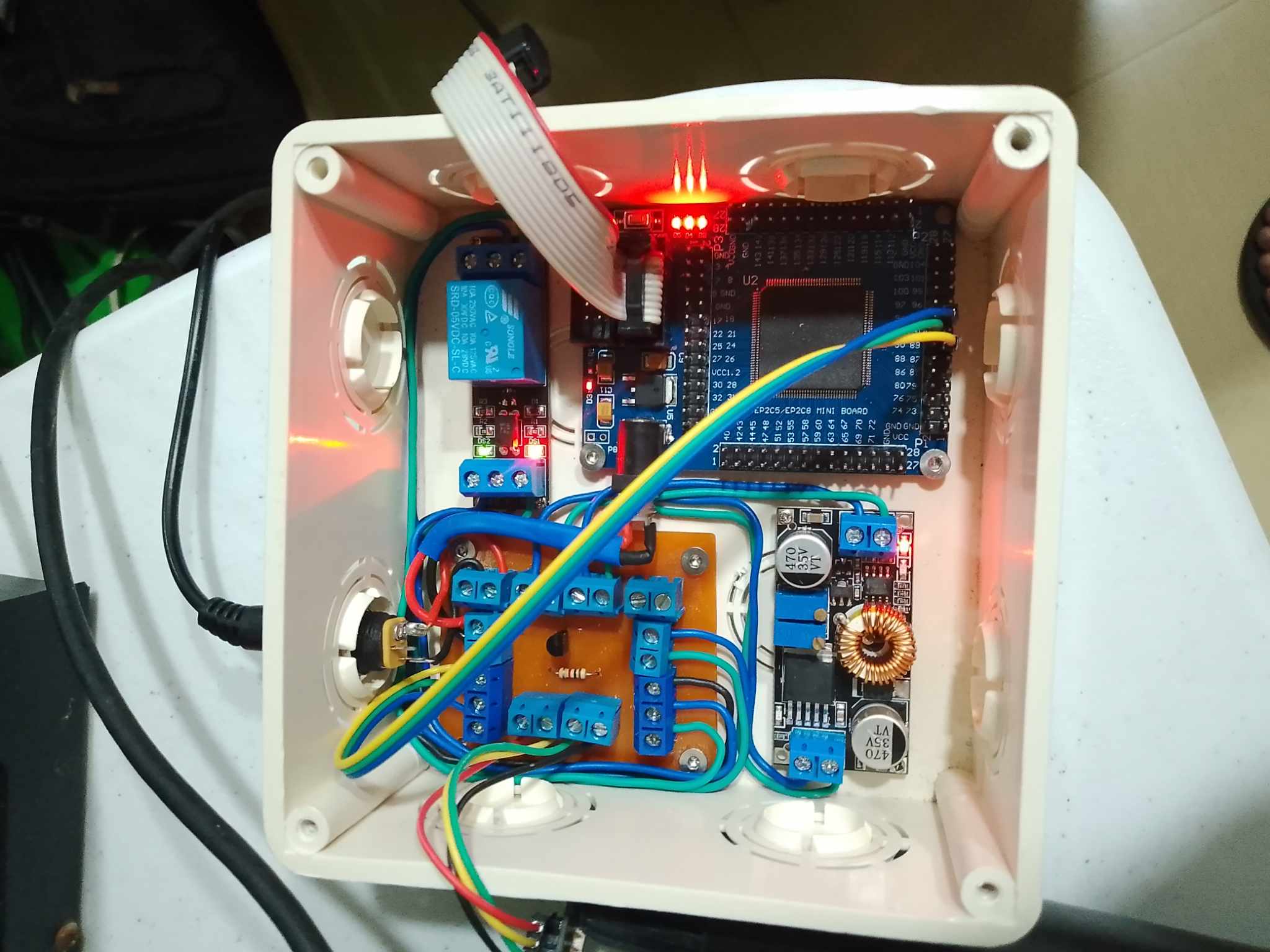
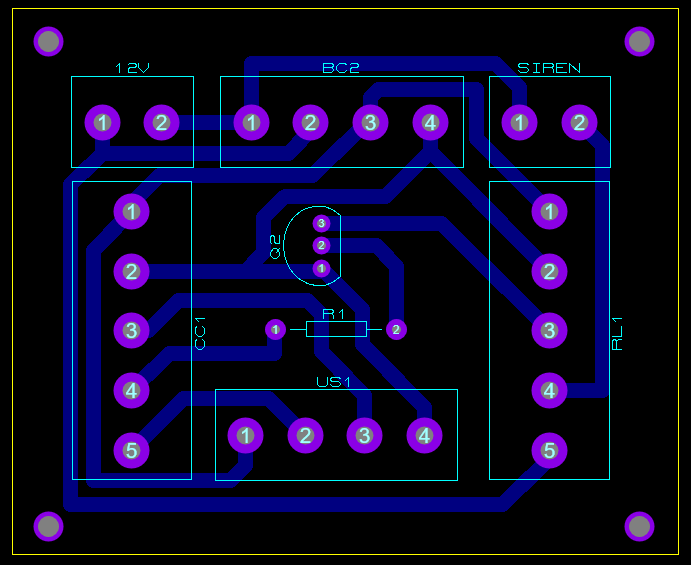
end process;

end Behavioral;

**PIN CONFIGURATION (FPGA Cyclone 2)**



**PCB LAYOUT**



The device is powered by a 12V 5A power supply. FPGA Cyclone 2 is used as a microcontroller for the system to function by uploading a series of codes.



The PVC pipe is used to narrow down the sensor’s width of detection. The receiver part of the sensor is where the PVC pipe is installed while the transmitter is laid bare.