Marco Pellis

NM IN1800051

# Relazione del progetto finale di Sistemi Integrati su FPGA

# “Rilevatore di posizione ad ultrasuoni”

## Introduzione

Lo scopo del progetto era realizzare un sistema di rilevamento di posizione utilizzando dei sensori ad ultrasuoni ed un’interfaccia che permettesse di vedere la posizione dell’oggetto su un piano xy. Per raggiungere questo obiettivo sono stati utilizzati:

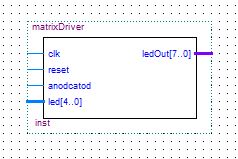
* una scheda Altera DE1, in congiunzione al tool di sviluppo **Quartus® II, versione 9.1 Build 350**
* dei sensori ultrasonici *HC-SR04*
* una matrice Led 8x8.

La scheda DE1 controlla i sensori e, ricevute da questi ultimi le distanze corrispondenti, le elabora attraverso formule matematiche fino a trovare la posizione dell’oggetto in xy e successivamente la trasporta sulla matrice led.

## Linguaggio di programmazione scelto

Inizialmente si era deciso di sviluppare il lavoro in C, sfruttando la possibilità di usare la scheda DE1 come un processore tramite Qsys. Il vantaggio di tale scelta sarebbe stato la possibilità di utilizzare codice già esistente per il controllo dei sensori, modificandolo opportunamente. Purtroppo non si è potuto sfruttare questa soluzione, in quanto, per motivi sconosciuti, il programma *Altera Monitor Program* non funzionava correttamente, dando un errore al momento del caricamento nel processore del programma scritto in C. Dopo svariati tentativi si è scelto di utilizzare il linguaggio Verilog per lo sviluppo dell’intero progetto.

## Driver della Matrice Led



Si è iniziato partendo dallo sviluppo del driver della matrice led. Si è creato il modulo *matrixDriver* che prende in ingresso, oltre a reset e clock, il valore di x o y ed un booleano che dica se il blocco si riferisce al catodo o all’anodo; in uscita si ha la porta GPIO\_1, con gli opportuni indirizzi. I pin vengono utilizzati come anodo o catodo della matrice led. Subito a monte è stato realizzato il modulo *testFor*[[1]](#footnote-1) per differenziare il blocco sopradescritto in anodo o in catodo: grazie a questo modulo è possibile aggiungere, nel caso che la matrice sia bicolore, gli anodi relativi all’altro colore. Per questo progetto ci si è serviti solo di un colore, di conseguenza le uscite per gli anodi rossi sono state commentate.

A monte di quest’ultimo modulo c’è il modulo *converter*, che determina gli intervalli di accensione per led. Questo modulo verrà discusso nel paragrafo relativo al calcolo di x e y.

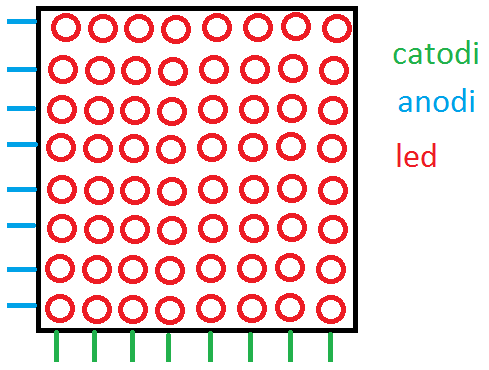
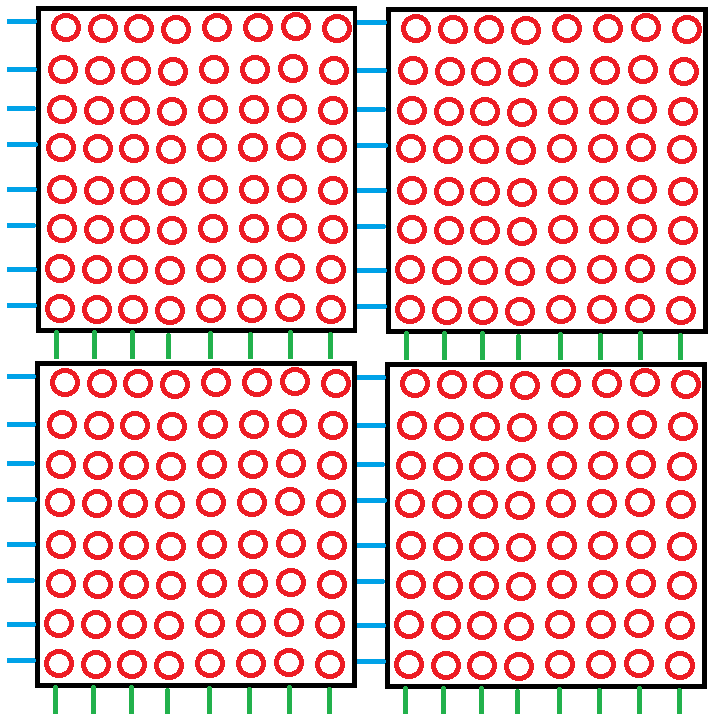
La matrice utilizzata è una matrice led 8x8 bicolore. Unendo opportunamente più matrici (in serie come in *Figura 1: matrici led in serie*) è possibile controllare la matrice più grande risultante, semplicemente modificando gli intervalli di distanza per led ed aggiungendo i casi necessari a ottenere il giusto numero di righe e colonne (anodi e catodi). In questo modo è possibile ottenere una migliore risoluzione della posizione dell’oggetto.

Figura 1: matrici led in serie

Figura 2: matrice Led

## Sensori ad ultrasuoni

Come già detto nell’introduzione, si sono scelti i sensori ad ultrasuoni HC-SR04, a causa del loro rapporto qualità/prezzo nettamente superiore. Si sono fatti brevi test con gli *SRF05*, ma le prestazioni non si differenziavano molto da quelle degli *SR04*: la differenza Si è riscontrata soprattutto nell’aumento di range di un metro e nella presenza di un piedino in più che svolgeva un compito non utile a questo progetto. È stato preso in considerazione anche il sensore *SRF08*, ma il costo è risultato eccessivo. Si è dunque scelto il sensore *HC-SR04*[[2]](#footnote-2)*.*

Utilizzando il datasheet[[3]](#footnote-3) come riferimento, si è sviluppato il modulo *ranging module* per calcolare la distanza tra sensore ed oggetto da misurare. Questo modulo manda al sensore un segnale TTL di trigger (di almeno 10µs come da datasheet) e calcola la differenza di tempo tra la fine del segnale di trigger e la ricezione dell’eco, ed infine trasforma questo tempo in distanza. La distanza, in uscita dal modulo, è data con la precisione del centimetro, ma basta commentare l’ultima riga del modulo per raggiungere una risoluzione al millimetro.

Il modulo presenta dei parametri ad hoc per l’esperimento, quali *maxDistance* e *calibrationEcho*. Il primo parametro serve a limitare la distanza di rilevazione oltre la quale il modulo segnala il superamento della distanza massima, e la scheda accende alcuni led rossi (spiegazione in seguito). Il secondo parametro serve a calibrare il segnale di eco ed è calcolato secondo le indicazioni del datasheet.

## Input e output della scheda

### Led Rossi

Dato che per determinare la posizione di un oggetto servono almeno tre sensori, sono stati utilizzati tre ranging\_module, e ad ogni sensore sono stati associati tre led rossi sulla scheda, che si accendono quando viene superata la distanza massima.

### 7Seg e Switch

I moduli a sette segmenti sono stati utilizzati per visualizzare le diverse distanze. Ad ogni sensore è associato uno switch Se si modifica lo stato dello switch, lo schermo riporterà la distanza misurata (in centimetri) dal relativo sensore. Al primo sensore è associato il primo switch. È possibile visualizzare una sola distanza alla volta: ogni switch risulta non funzionante se uno switch con un numero più basso è attivo. Per controllare la matrice sono stati realizzati due blocchi che fungono da decoder per indirizzare unità, decine, centinaia e migliaia nel segmento giusto.

### Key

Il Key[0] è stato adibito a reset.

## Disposizione sensori e problemi relativi

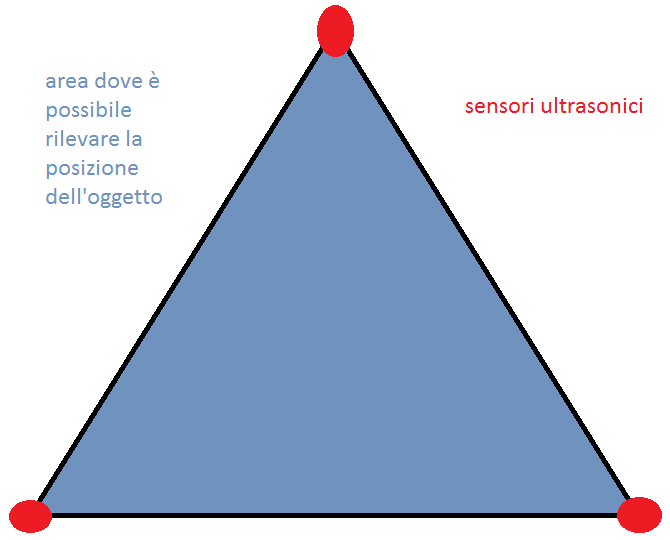
La parte centrale del progetto è stata la conversione delle distanze in coordinate xy. Per far ciò è stato necessario sviluppare parecchi moduli. Il risultato voluto, e raggiunto, era l’illuminazione del Led corrispondente alla posizione dell’oggetto o l’illuminazione di tutta la matrice nel caso in cui l’oggetto in questione fosse all’esterno dell’area.

Figura 3: posizionamento sensori e area di interesse

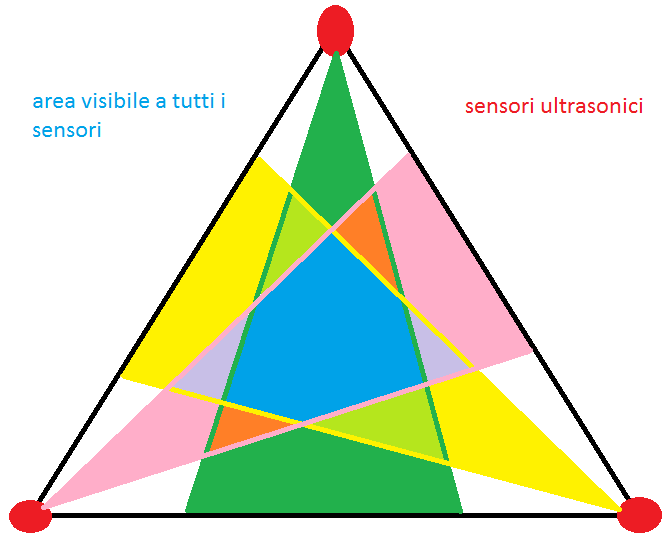
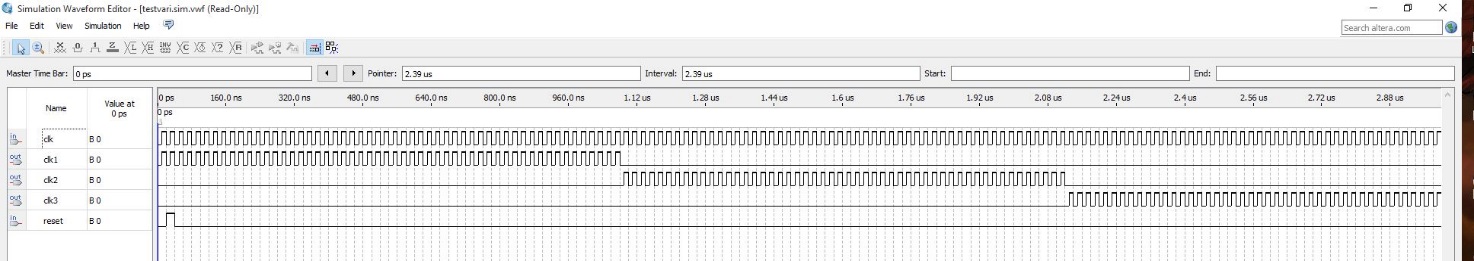
Per determinare la posizione di un oggetto, si ha bisogno di tre sensori ultrasonici. Si è scelto di posizionare i sensori ai vertici di un triangolo equilatero e delimitare l’area sensibile al triangolo racchiuso da questi tre sensori. La distanza massima per ogni sensore è stata impostata come la lunghezza del lato del triangolo. Se la rilevazione risulta esterna all’area del triangolo, la matrice Led si illuminerà tutta, confermando che l’oggetto preso in esame è fuori dall’area stabilita per la rilevazione.

Figura 4 : area di interesse reale tre sensori

Tuttavia, le caratteristiche dei sensori hanno dato luogo a due problemi. Il primo riguarda il fatto che i sensori non hanno un angolo di visuale di 60 gradi, ma funzionano in maniera ottimale in un intervallo di 30 gradi ed anno un angolo massimo di 45 gradi, come descritto nel manuale d’uso del componente[[4]](#footnote-4). L’area di rilevazione, quindi, non è tutto il triangolo ma solo una porzione (centrale se i sensori sono posizionati precisamente) del triangolo. Questo problema può essere evitato delimitando ulteriormente l’area di rilevazione, oppure aggiungendo sensori, come si potrà vedere nella sezione *“Altre caratteristiche del progetto*”. La soluzione adottata in questo caso è stata la prima, e l’area considerata per la rilevazione risulterà essere più piccola.

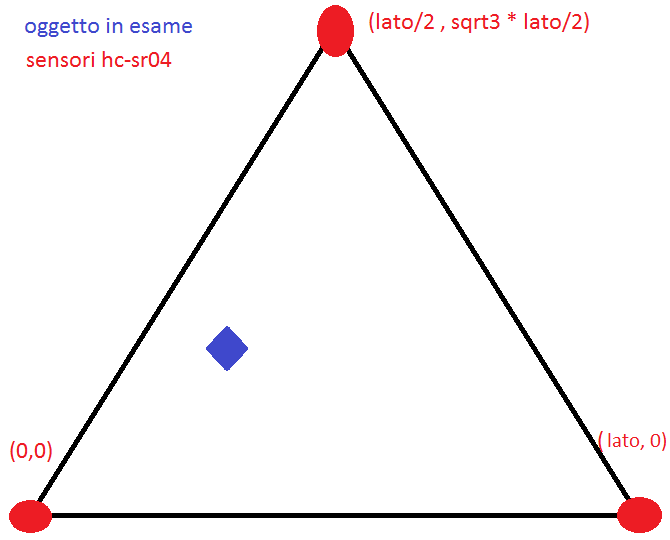
Il secondo problema è rappresentato dal fatto che gli ultrasuoni emessi da un sensore vengono percepiti dagli altri sensori, quindi la massima distanza rilevata sarà sempre metà della lunghezza massima[[5]](#footnote-5). Si è dunque creato un modulo che alterna il segnale di clock ai tre sensori. Quando il primo sensore è in funzione, gli altri due sensori sono disabilitati e quindi non calcolano la distanza. Dopo un tempo opportuno il clock entrante nel sensore 1 si interrompe e parte il clock entrante nel sensore 2, poi si passa al 3, per poi ricominciare dal sensore 1.

Come mostrato nella figura, il clock del primo sensore si interrompe ad un numero preciso di iterazioni dato definito nel blocco *main*, per passare al sensore successivo.

## Calcolo delle coordinate

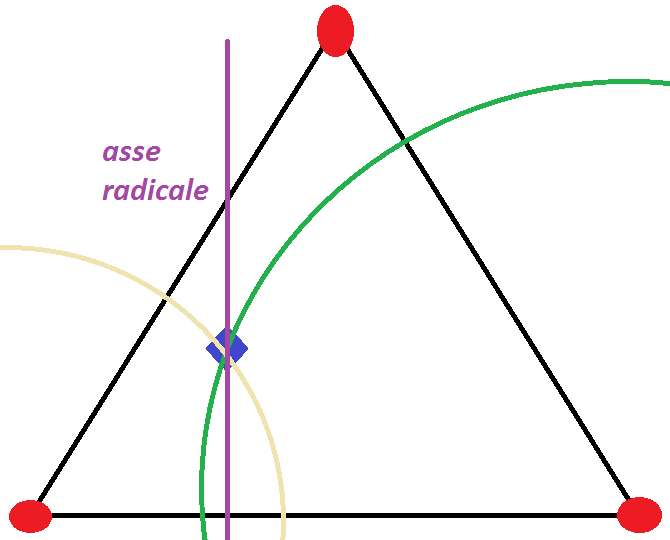
Il calcolo di x e y avviene tramite il calcolo dell’asse radicale[[6]](#footnote-6) e l’utilizzo del teorema di Pitagora.

Si parte dal calcolare la coordinata x tramite l’asse radicale, per poi passare a calcolare la y tramite il teorema Pitagora.

Per prima cosa si fissano i tre punti relativi ai sensori:

* Sensore 1: (0,0)
* Sensore 2: (lato,0)
* Sensore 3: (lato/2, sqrt3 \* lato/2)

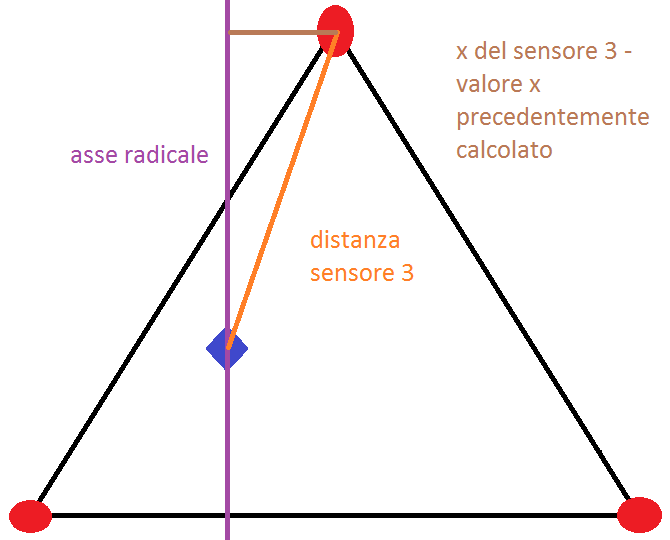
dove il lato è dato dalla distanza tra due sensori (dato che il triangolo preso in esame è costruito equilatero tutte le distanze saranno uguali) e sqrt3 è √3. Per semplicità e per il fatto che sono necessari numeri interi, vengono inseriti direttamente dei numeri arrotondati come dati iniziali.

I sensori forniscono le distanze dall’oggetto in esame, che non sono altro che i raggi delle circonferenze centrate nel sensore e passanti per il punto dove l’oggetto è posizionato. Le due circonferenze hanno due punti in comune e l’asse radicale è la retta che passa per quei due punti. Avendo scelto i due sensori alla base del triangolo come asse delle x, l’asse radicale sarà una retta perpendicolare all’asse delle x e darà un’equazione del tipo:

La formula dell’asse radicale è la seguente:

Con

dove r’ è il raggio della prima circonferenza, ovvero la distanza rilevata dal primo sensore, r” è il raggio della seconda circonferenza, α’, β’, α” e β” sono le coordinate dei centri delle circonferenze (posizioni del primo e del secondo sensore). Inserendo i valori della posizione del sensore si ottiene:

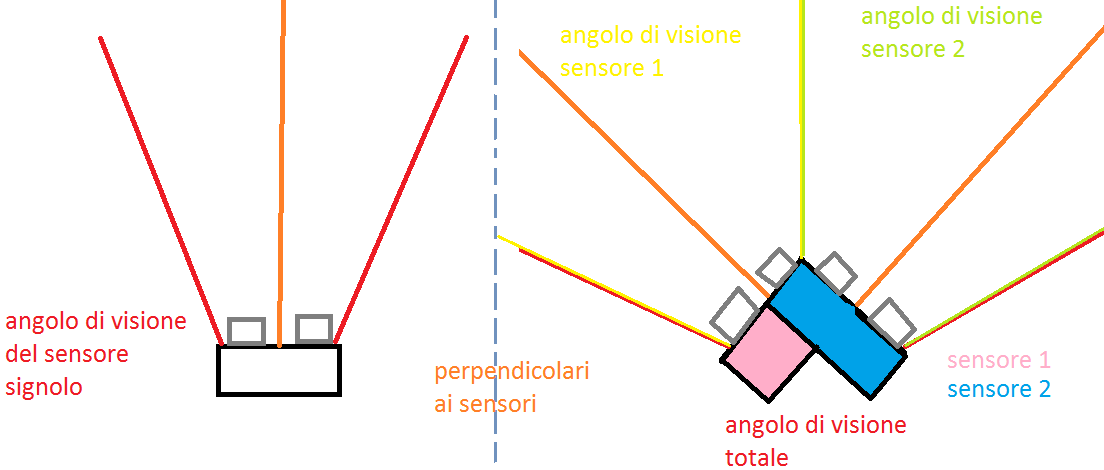
Per il calcolo di y, si può individuare nella figura un triangolo rettangolo di cui si conoscono due lati tramite il teorema di Pitagora è possibile calcolare il terzo lato, che sottratto all’altezza massima possibile (la y della posizione del sensore 3), ci darà la posizione sull’asse delle y del nostro oggetto.

Si sono dunque ottenute le coordinate x e y dell’oggetto: questi parametri andranno passati al programma di controllo della matrice led che permetterà di visualizzare la posizione.

Si è scelto questo approccio, cioè il calcolo della x tramite l’asse radicale, in modo tale da avere sempre una soluzione all’equazione. Se l’oggetto fosse molto grande, infatti, non si avrebbe sempre una soluzione dell’intersezione delle due circonferenze oppure la si potrebbe avere imprecisa. Con l’asse radicale si ottiene sempre la x del centro dell’oggetto preso in esame.

La problematica maggiore di questa parte è stato il calcolo della radice quadrata, in quanto si possono utilizzare soltanto numeri interi. Si è dovuto implementare un modulo esclusivamente per questo scopo. In questo modulo compare un valore chiamato *count* che aumenta ad ogni ciclo finché *count\*count* non supera il valore al quadrato. A quel punto in uscita si avrà il valore *count–1*.

## Altre caratteristiche del progetto

Il progetto ha una parte commentata che permette di ampliare l’area per il rilevamento dell’oggetto in questione. Come già detto nel capitolo “Disposizione sensori e problemi relativi”, a causa del ridotto angolo dei sensori, l’area utilizzabile per la rilevazione si riduce di molto. Togliendo dai commenti la giusta parte di codice e inserendo le giuste porte I/O è possibile aggiungere sensori al progetto, allargando il cono di funzionamento ottimale da 30 a 60 gradi, in modo da coprire tutto il triangolo.

La parte di codice commentata fa partire insieme la rilevazione da parte dei due sensori, calcola la distanza relativa ad entrambi per poi restituire la distanza minore[[7]](#footnote-7).

## Note

Il progetto è stato realizzato per rilevare la posizione di un solo oggetto all’interno di un’area definita. Nel caso di più oggetti all’interno dell’area, si avrà come valore delle x il punto medio del segmento immaginario che unisce i due oggetti più vicini ai sensori posizionati alla base del triangolo [in (0,0) e in (lato,0)], mentre come valore delle y la distanza minore dell’oggetto posizionato vicino al sensore più in alto [in (lato/2, sqrt3\*lato/2)]. Questo vale sia per la disposizione a 3 sensori che per quella a 6 sensori.

Appendice 1

È riportato di seguito il codice con le modifiche da fare nel caso si vogliano usare sei sensori oppure una matrice led più grande.

#### Main (originale)

module main(

// clock

input CLOCK\_50,

// input & output

input wire [35:0] GPIO\_0,

input [9:0] SW,

input [3:0] KEY,

output wire [35:0] GPIO\_1,

output [6:0] HEX0, HEX1, HEX2, HEX3,

output [7:0] LEDG,

output [9:0] LEDR

);

// Parameter sensors

parameter side = 68;

parameter xSensor1st = 0; // origin

parameter xSensor2nd = side; // side

parameter xSensor3rd = xSensor2nd / 2; // side/2

parameter ySensor1st = 0; // origin

parameter ySensor2nd = 0; // y=0

parameter ySensor3rd = 62; // ~sqrt(3)\*side/2 (case:equilateral triangle)

parameter maxDistanceSensors = xSensor2nd; // max distance

parameter minDistanceSensors = 0; // min distance

// Parameter matrix

parameter column = 8;

parameter row = 8;

// Key block

wire key\_reset;

assign reset = ~KEY[0];

// address

assign echo1 = GPIO\_0[0]; //sensors

assign GPIO\_1[33] = trigger1;

assign echo2 = GPIO\_0[1];

assign GPIO\_1[34] = trigger2;

assign echo3 = GPIO\_0[2];

assign GPIO\_1[35] = trigger3;

wire [32:0] Linee; //Led Matrix

assign GPIO\_1 [32:0] = Linee;

//wire

wire [11:0] distance;

wire [11:0] distance1;

wire [11:0] distance2;

wire [11:0] distance3;

wire clk1;

wire clk2;

wire clk3;

wire [31:0] control;

diffClockComplete( // to distinguish sensor

.clk(CLOCK\_50),

.reset(reset),

.control(750000),

.clk1(clk1),

.clk2(clk2),

.clk3(clk3)

);

// modules to calculate the three distances

ranging\_module rangeFirstA( //1st sensor

.clk(clk1),

.echo(echo1),

.reset(reset),

.collegamenti(Linee),

.period\_cnt\_full\_out(control),

.trig(trigger1),

.distance(distance1) //distance1a (2 sensors for angle)

);

ranging\_module rangeSecondA( //2nd sensor

.clk(clk2),

.echo(echo2),

.reset(reset),

.collegamenti(Linee),

.trig(trigger2),

.distance(distance2) //distance2a (2 sensors for angle)

);

ranging\_module rangeThirdA( //3rd sensor

.clk(clk3),

.echo(echo3),

.reset(reset),

.collegamenti(Linee),

.trig(trigger3),

.distance(distance3) //distance3a (2 sensors for angle)

);

// modules if every angle have 2 sensor (problem of range)

/\*

ranging\_module rangeFirstB( //1st sensor

.clk(clk1),

.echo(echo1),

.reset(reset),

.collegamenti(Linee),

.period\_cnt\_full\_out(control),

.trig(trigger1),

.distance(distance1b)

);

ranging\_module rangeSecondB( //2nd sensor

.clk(clk2),

.echo(echo2),

.reset(reset),

.collegamenti(Linee),

.trig(trigger2),

.distance(distance2b)

);

ranging\_module rangeThirdB( //3rd sensor

.clk(clk3),

.echo(echo3),

.reset(reset),

.collegamenti(Linee),

.trig(trigger3),

.distance(distance3b)

);

chooseOne sensor1st(

.clk(clk1),

.reset(reset),

.distanceA(distance1a),

.distanceB(distance1b),

.distanceMinimum(distance1)

);

chooseOne sensor2nd(

.clk(clk1),

.reset(reset),

.distanceA(distance2a),

.distanceB(distance2b),

.distanceMinimum(distance2)

);

chooseOne sensor3rd(

.clk(clk1),

.reset(reset),

.distanceA(distance3a),

.distanceB(distance3b),

.distanceMinimum(distance3)

);

\*/

// from distances to LedMatrix Chain

xCalc test5(

.clk(CLOCK\_50),

.reset(reset),

.sens1(distance1), //.sens1(distance1 - minDistanceSensors),

.sens2(distance2), //.sens2(distance2 - minDistanceSensors),

.sens3(distance3), //.sens3(distance3 - minDistanceSensors),

.xSensor1(xSensor1st),

.xSensor2(xSensor2nd),

.xSensor3(xSensor3rd),

.ySensor1(ySensor1st),

.ySensor2(ySensor2nd),

.ySensor3(ySensor3rd),

.maxDistance(maxDistanceSensors),

.minDistance(minDistanceSensors), //not used

.numberColumn(column),

.numberRow(row),

.Linee(Linee)

);

// to switch which distance show

switch7seg switch(

.clk(CLOCK\_50),

.reset(reset),

.distance1(distance1),

.distance2(distance2),

.distance3(distance3),

.Switch(SW),

.distance(distance)

);

// BCD convertion - decoder

wire [3:0] distance\_ones;

wire [3:0] distance\_tens;

wire [3:0] distance\_hundreds;

wire [3:0] distance\_thousands;

bin2bcd decoder(

.B({2'b00, distance}),

.BCD\_0(distance\_ones),

.BCD\_1(distance\_tens),

.BCD\_2(distance\_hundreds),

.BCD\_3(distance\_thousands)

);

// Show distance on HEX block

hex\_7seg decoder2(

.cs(distance\_ones),

.ds(distance\_tens),

.s(distance\_hundreds),

.das(distance\_thousands),

.seg0(HEX0),

.seg1(HEX1),

.seg2(HEX2),

.seg3(HEX3)

);

// LedR ON if distance out of limit

Limit ifTooLong( //3 leds open for sensor out of range

.clk(CLOCK\_50),

.distance1(distance1), //1,2,3

.distance2(distance2), //4,5,6

.distance3(distance3), //7,8,9

.led(LEDR)

);

Endmodule

##### Main (modificato)

module main(

// clock

input CLOCK\_50,

// input & output

input wire [35:0] GPIO\_0,

input [9:0] SW,

input [3:0] KEY,

output wire [35:0] GPIO\_1,

output [6:0] HEX0, HEX1, HEX2, HEX3,

output [7:0] LEDG,

output [9:0] LEDR

);

// Parameter sensors

parameter side = 68;

parameter xSensor1st = 0; // origin

parameter xSensor2nd = side; // side

parameter xSensor3rd = xSensor2nd / 2; // side/2

parameter ySensor1st = 0; // origin

parameter ySensor2nd = 0; // y=0

parameter ySensor3rd = 62; // ~sqrt(3)\*side/2 (case:equilateral triangle)

parameter maxDistanceSensors = xSensor2nd; // max distance

parameter minDistanceSensors = 0; // min distance

// Parameter matrix

parameter column = 8;

parameter row = 8;

// Key block

wire key\_reset;

assign reset = ~KEY[0];

// address

assign echo1 = GPIO\_0[0]; //sensors a

assign GPIO\_1[33] = trigger1;

assign echo2 = GPIO\_0[1];

assign GPIO\_1[34] = trigger2;

assign echo3 = GPIO\_0[2];

assign GPIO\_1[35] = trigger3;

assign echo1 = GPIO\_0[3]; //sensors b

assign GPIO\_1[30] = trigger1;

assign echo2 = GPIO\_0[4];

assign GPIO\_1[31] = trigger2;

assign echo3 = GPIO\_0[5];

assign GPIO\_1[32] = trigger3;

wire [32:0] Linee; //Led Matrix

assign GPIO\_1 [32:0] = Linee;

//wire

wire [11:0] distance;

wire [11:0] distance1a;

wire [11:0] distance2a;

wire [11:0] distance3a;

wire [11:0] distance1b;

wire [11:0] distance2b;

wire [11:0] distance3b;

wire [11:0] distance1;

wire [11:0] distance2;

wire [11:0] distance3;

wire clk1;

wire clk2;

wire clk3;

wire [31:0] control;

diffClockComplete( // to distinguish sensor

.clk(CLOCK\_50),

.reset(reset),

.control(750000),

.clk1(clk1),

.clk2(clk2),

.clk3(clk3)

);

// modules to calculate the three distances

ranging\_module rangeFirstA( //1st sensor

.clk(clk1),

.echo(echo1),

.reset(reset),

.collegamenti(Linee),

.period\_cnt\_full\_out(control),

.trig(trigger1),

.distance(distance1a)

);

ranging\_module rangeSecondA( //2nd sensor

.clk(clk2),

.echo(echo2),

.reset(reset),

.collegamenti(Linee),

.trig(trigger2),

.distance(distance2a)

);

ranging\_module rangeThirdA( //3rd sensor

.clk(clk3),

.echo(echo3),

.reset(reset),

.collegamenti(Linee),

.trig(trigger3),

.distance(distance3a)

);

// modules if every angle have 2 sensor (problem of range)

ranging\_module rangeFirstB( //1st sensor

.clk(clk1),

.echo(echo1),

.reset(reset),

.collegamenti(Linee),

.period\_cnt\_full\_out(control),

.trig(trigger1),

.distance(distance1b)

);

ranging\_module rangeSecondB( //2nd sensor

.clk(clk2),

.echo(echo2),

.reset(reset),

.collegamenti(Linee),

.trig(trigger2),

.distance(distance2b)

);

ranging\_module rangeThirdB( //3rd sensor

.clk(clk3),

.echo(echo3),

.reset(reset),

.collegamenti(Linee),

.trig(trigger3),

.distance(distance3b)

);

chooseOne sensor1st(

.clk(clk1),

.reset(reset),

.distanceA(distance1a),

.distanceB(distance1b),

.distanceMinimum(distance1)

);

chooseOne sensor2nd(

.clk(clk1),

.reset(reset),

.distanceA(distance2a),

.distanceB(distance2b),

.distanceMinimum(distance2)

);

chooseOne sensor3rd(

.clk(clk1),

.reset(reset),

.distanceA(distance3a),

.distanceB(distance3b),

.distanceMinimum(distance3)

);

// from distances to LedMatrix Chain

xCalc test5(

.clk(CLOCK\_50),

.reset(reset),

.sens1(distance1 - minDistanceSensors),

.sens2(distance2 - minDistanceSensors),

.sens3(distance3 - minDistanceSensors),

.xSensor1(xSensor1st),

.xSensor2(xSensor2nd),

.xSensor3(xSensor3rd),

.ySensor1(ySensor1st),

.ySensor2(ySensor2nd),

.ySensor3(ySensor3rd),

.maxDistance(maxDistanceSensors),

.minDistance(minDistanceSensors), //not used

.numberColumn(column),

.numberRow(row),

.Linee(Linee)

);

// to switch which distance show

switch7seg switch(

.clk(CLOCK\_50),

.reset(reset),

.distance1(distance1),

.distance2(distance2),

.distance3(distance3),

.Switch(SW),

.distance(distance)

);

// BCD convertion - decoder

wire [3:0] distance\_ones;

wire [3:0] distance\_tens;

wire [3:0] distance\_hundreds;

wire [3:0] distance\_thousands;

bin2bcd decoder(

.B({2'b00, distance}),

.BCD\_0(distance\_ones),

.BCD\_1(distance\_tens),

.BCD\_2(distance\_hundreds),

.BCD\_3(distance\_thousands)

);

// Show distance on HEX block

hex\_7seg decoder2(

.cs(distance\_ones),

.ds(distance\_tens),

.s(distance\_hundreds),

.das(distance\_thousands),

.seg0(HEX0),

.seg1(HEX1),

.seg2(HEX2),

.seg3(HEX3)

);

// LedR ON if distance out of limit

Limit ifTooLong( //3 leds open for sensor out of range

.clk(CLOCK\_50),

.distance1(distance1), //1,2,3

.distance2(distance2), //4,5,6

.distance3(distance3), //7,8,9

.led(LEDR)

);

Endmodule

##### Converter

module converter(

input wire clk,

input wire reset,

input wire [11:0] numeroX,

input wire [11:0] numeroY,

input wire [11:0] ySensor3,

input wire [11:0] maxDistance,

input wire [11:0] minDistance,

input wire [4:0] numberRow,

input wire [4:0] numberColumn,

output wire [4:0] ledC,

output wire [4:0] ledAG,

output wire [35:0] Linee

);

// wire [35:0] Linee;

reg [11:0] xValue;

reg [11:0] yValue;

reg [11:0] xLed;

reg [11:0] yLed; //

always@ (posedge clk or posedge reset)

begin

if (reset)

begin

xValue <= 4'b0000;

yValue <= 4'b0000;

end

else

begin

xLed <= (maxDistance - minDistance) / numberRow; //range

yLed <= (ySensor3 - minDistance) / numberColumn;

if (numeroX <= xLed)

xValue <= 4'b0001;

else if (numeroX > xLed && numeroX <= (xLed\*2))

xValue <= 4'b0010;

else if (numeroX > (xLed \*2) && numeroX <= (xLed \* 3))

xValue <= 4'b0011;

else if (numeroX > (xLed \*3) && numeroX <= (xLed \* 4))

xValue <= 4'b0100;

else if (numeroX > (xLed \*4) && numeroX <= (xLed \* 5))

xValue <= 4'b0101;

else if (numeroX > (xLed \*5) && numeroX <= (xLed \* 6))

xValue <= 4'b0110;

else if (numeroX > (xLed \*6) && numeroX <= (xLed \* 7))

xValue <= 4'b0111;

else if (numeroX > (xLed \*7) && numeroX <= (xLed \* 8))

xValue <= 4'b1000;

if (numeroY <= yLed)

yValue <= 4'b0001;

else if (numeroY > yLed && numeroY <= (yLed\*2))

yValue <= 4'b0010;

else if (numeroY > (yLed \*2) && numeroY <= (yLed \* 3))

yValue <= 4'b0011;

else if (numeroY > (yLed \*3) && numeroY <= (yLed \* 4))

yValue <= 4'b0100;

else if (numeroY > (yLed \*4) && numeroY <= (yLed \* 5))

yValue <= 4'b0101;

else if (numeroY > (yLed \*5) && numeroY <= (yLed \* 6))

yValue <= 4'b0110;

else if (numeroY > (yLed \*6) && numeroY <= (yLed \* 7))

yValue <= 4'b0111;

else if (numeroY > (yLed \*7) && numeroY <= (yLed \* 8))

yValue <= 4'b1000;

if (numeroX > (xLed \*8)||numeroY > (yLed \*8))

begin

xValue <= 4'b1001;

yValue <= 4'b1001;

end

end

end

testFor toMatrix(.clk(clk), .numeroC(xValue), .numeroAG(yValue), .couple(Linee));

endmodule

negli if numeroX e numeroY si possono aggiungere altri casi per ampliare la matrice e rendere più precisa la risoluzione (ovviamente bisogna aggiungere sia le uscite nell’indirizzo uscita che aggiungere le matrici led fisiche e collegarle ai giusti pin).

Appendice 2

È riportato di seguito, per completezza, alcune parti di codice più importanti:

###### il clock dei sensori:

module diffClock(

//input

input wire clk,

input wire reset,

input wire [31:0] control,

//output

output reg clk1,

output reg clk2,

output reg clk3

);

reg [32:0] count; // counter - sovradimensioned for every "control"

reg [3:0] whichOutput; // to decide which sensor will be up

parameter cycleForSensor = 3; //2 // how many cycles for every sensor

always@(posedge clk or posedge reset)

begin

if (reset)

begin

count <= 0;

whichOutput <= 0;

end

else if (count <= control) // control = 500000

begin

count = count + 1;

if(whichOutput >= 0 && whichOutput < 1 \* cycleForSensor) //1st sensor up

begin

clk1 <= 1'b1;

clk2 <= 1'b0;

clk3 <= 1'b0;

end

else if(whichOutput >= 1 \* cycleForSensor && whichOutput < 2 \* cycleForSensor) //pause - wait all waves end

begin

clk1 <= 1'b0;

clk2 <= 1'b0;

clk3 <= 1'b0;

end

else if(whichOutput >= 2 \* cycleForSensor && whichOutput < 3 \* cycleForSensor) //2nd sensor up

begin

clk1 <= 1'b0;

clk2 <= 1'b1;

clk3 <= 1'b0;

end

else if(whichOutput >= 3 \* cycleForSensor && whichOutput < 4 \* cycleForSensor) //pause - wait all waves end

begin

clk1 <= 1'b0;

clk2 <= 1'b0;

clk3 <= 1'b0;

end

else if(whichOutput >= 4 \* cycleForSensor && whichOutput < 5 \* cycleForSensor) //3rd sensor up

begin

clk1 <= 1'b0;

clk2 <= 1'b0;

clk3 <= 1'b1;

end

else if(whichOutput >= 5 \* cycleForSensor && whichOutput < 6 \* cycleForSensor) //pause - wait all waves end

begin

clk1 <= 1'b0;

clk2 <= 1'b0;

clk3 <= 1'b0;

end

end

else if (count > control) // control = 500000

begin

count <= 0;

if (whichOutput < 6\*cycleForSensor) // changing output

whichOutput = whichOutput +1;

else

whichOutput <= 0;

end

end

endmodule

###### il driver dei sensori:

module ranging\_module(

input wire clk,

input wire echo,

input wire reset,

input wire [31:0] collegamenti,

output reg trig,

output wire [11:0] distance,

output wire [23:0] period\_cnt\_output,

output wire [31:0] period\_cnt\_full\_out

);

parameter maxDistance = 1010; //max distance

parameter calibrationEcho = 295; // calibration for a better

parameter period\_cnt\_full\_max =5000000;

assign period\_cnt\_full\_out = period\_cnt\_full\_max;

reg [23:0] period\_cnt;

wire period\_cnt\_full;

always @(posedge clk or posedge reset)

begin

if (reset)

period\_cnt <= 0;

else

begin

if (period\_cnt\_full)

begin

period\_cnt <= 0;

end

else

begin

period\_cnt <= period\_cnt + 1;

end

end

end

assign period\_cnt\_full = (period\_cnt == period\_cnt\_full\_max + 1);

// DEBUG

assign period\_cnt\_output = period\_cnt;

// Trigger di 100 microsec

always @(posedge clk or posedge reset)

begin

if (reset)

trig <= 0;

else

trig <= ( period\_cnt > 1000 && period\_cnt < 51000);

end

// Echo

reg [11:0] echo\_length;

always @(posedge clk or posedge reset)

begin

if (reset)

echo\_length <= 0;

else

begin

if (trig || echo\_length == calibrationEcho)

echo\_length <= 0;

else if (echo)

echo\_length <= echo\_length + 1;

end

end

// cap distance : combination distance-echo

reg [11:0] distance\_temp;

always @(posedge clk or posedge reset)

begin

if (reset)

distance\_temp <= 0;

else

begin

if (trig)

distance\_temp <= 0;

else if (echo && echo\_length == calibrationEcho && distance\_temp < maxDistance)//(echo && echo\_length == 275 && distance\_temp < 500)

distance\_temp <= distance\_temp + 1;

end

end

// output Distance

reg [11:0] distance\_output;

always @(posedge clk or posedge reset)

begin

if (reset)

distance\_output <= 0;

else if (period\_cnt\_full)

begin

distance\_output <= distance\_temp;

end

end

assign distance = distance\_output / 10; // (\*/10) for cm

endmodule

###### calcolo della radice quadrata:

module sqrtModule( //sqrt calc

input wire clk,

input wire reset,

input wire [11:0] xValue,

input wire [31:0] y1Value,

input wire [11:0] ySensor3,

input wire [11:0] maxDistance,

input wire [11:0] minDistance,

input wire [4:0] numberRow,

input wire [4:0] numberColumn,

output wire valueX,

output wire [11:0]valueSqrtY1,

output wire [35:0] Linee //GPIO\_1

);

reg [11:0] count;

reg [11:0] yProvv;

always@ (posedge clk or posedge reset)

begin

if (reset)

begin

yProvv <= 11'b00000000000;

count = 0;

end

else

begin

if (count \* count < y1Value)

count = count + 1;

else if (count \* count >= y1Value)

begin

yProvv <= count -1;

count = 0;

end

end

end

assign valueSqrtY1 = yProvv;

yCalc goToRealY(.clk(clk), .reset(reset), .sqrtY1Value(yProvv), .xValue(xValue), .ySensor3(ySensor3), .maxDistance(maxDistance), .minDistance(minDistance), .numberRow(numberRow), .numberColumn(numberColumn), .Linee(Linee));

endmodule

###### Il calcolo di x:

module xCalc(

input wire clk,

input wire reset,

input wire [11:0] sens1, //calc sensor1

input wire [11:0] sens2, //calc sensor2

input wire [11:0] sens3, //calc sensor3

input wire [11:0] xSensor1, //positioning sensor center

input wire [11:0] xSensor2,

input wire [11:0] xSensor3,

input wire [11:0] ySensor1,

input wire [11:0] ySensor2,

input wire [11:0] ySensor3,

input wire [11:0] maxDistance, // max distance

input wire [11:0] minDistance, // min distance

input wire [4:0] numberRow,

input wire [4:0] numberColumn,

output wire [11:0] axisX,

// output wire [11:0] axisY,

output wire [35:0] Linee //GPIO\_1

);

reg [11:0] xValue;

// reg [11:0] yValue;

//value

wire [11:0] a1;

wire [11:0] b1;

wire [11:0] r1;

wire [11:0] a2;

wire [11:0] b2;

wire [11:0] r2;

wire [11:0] c1;

wire [11:0] c2;

reg [11:0] valueX;

assign a1 [11:0] = xSensor1;

assign b1 [11:0] = ySensor1;

assign r1 [11:0] = sens1;

assign a2 [11:0] = xSensor2;

assign b2 [11:0] = ySensor2;

assign r2 [11:0] = sens2;

assign c1 [11:0] = (a1 \* a1) + (b1 \* b1) - (r1 \* r1);

assign c2 [11:0] = (a2 \* a2) + (b2 \* b2) - (r2 \* r2);

always@ (posedge clk or posedge reset)

begin

if (reset)

begin

valueX <= 11'b00000000000;

end

else if (sens1 > maxDistance || sens2 > maxDistance) //out limit distance

begin

valueX <= 11'b11111111111;

end

else

begin

valueX <= ((((sens1)\*(sens1))-((sens2)\*(sens2)) +((xSensor2)\*(xSensor2)))/(2\*(xSensor2))); //calculate radical axis

end

end

assign axisX [11:0] = valueX;

y1Calc andNowY(

.clk(clk),

.reset(reset),

.sens3(sens3),

.xValue(valueX),

.ySensor3(ySensor3),

.xSensor3(xSensor3),

.maxDistance(maxDistance),

.minDistance(minDistance),

.numberRow(numberRow),

.numberColumn(numberColumn),

.Linee(Linee)

);

endmodule

###### il calcolo di y’ (=y^2):

module y1Calc(

input wire clk,

input wire reset,

input wire [11:0] xValue,

input wire [11:0] sens3,

input wire [11:0] xSensor3,

input wire [11:0] ySensor3,

input wire [11:0] maxDistance, // max distance between sensors

input wire [11:0] minDistance, // min distance from sensor

input wire [4:0] numberRow,

input wire [4:0] numberColumn,

output wire valueX,

output wire [31:0]valueY1,

output wire [35:0] Linee //GPIO\_1

);

reg [11:0] xProvv;

reg [31:0] yProvv;

always@ (posedge clk or posedge reset)

begin

if (reset)

begin

xProvv <= 11'b00000000000;

yProvv <= 11'b00000000000;

end

else if (sens3 > maxDistance)

begin

xProvv <= 11'b11111111111;

yProvv <= 11'b11111111111;

end

else

begin

/\*if (xValue == xSensor3)

begin

yProvv <= sens3;

xProvv <= xValue;

end\*/

//else

if (xValue >= xSensor3) // >= -> >

begin

yProvv <= ((sens3)\*(sens3)) + ((xValue - xSensor3)\*(xValue - xSensor3));

xProvv <= xValue;

end

else if (xValue < xSensor3)

begin

yProvv <= ((sens3)\*(sens3)) + ((xSensor3 - xValue)\*(xSensor3 - xValue));

xProvv <= xValue;

end

end

end

assign valueX = xProvv;

assign valueY1 = yProvv;

sqrtModule giveMeTheSqrt(

.clk(clk),

.reset(reset),

.y1Value(yProvv),

.xValue(xProvv),

.ySensor3(ySensor3),

.maxDistance(maxDistance),

.numberRow(numberRow),

.numberColumn(numberColumn),

.Linee(Linee)

);

Endmodule

Il codice completo si può trovare all’indirizzo di Github:  
<https://github.com/Inud/Position-Detector-HCSR04-Verilog-DE1>

1. Serve esclusivamente per assegnare gli indirizzi di GPIO\_1. [↑](#footnote-ref-1)
2. Il codice in ogni caso funziona anche per SRF05. [↑](#footnote-ref-2)
3. Datasheet http://www.micropik.com/PDF/HCSR04.pdf [↑](#footnote-ref-3)
4. https://docs.google.com/document/d/1Y-yZnNhMYy7rwhAgyL\_pfa39RsB-x2qR4vP8saG73rE/edit [↑](#footnote-ref-4)
5. Il calcolo della distanza è dato dal tempo che l’onda ha per andare all’oggetto e tornare dall’oggetto. Il secondo sensore però riceve direttamente l’onda generato da uno degli altri sensori, quindi il percorso dell’onda non risulta come andata e ritorno, ma solamente andata… e dato che nel calcolo della distanza compare una divisione per due in quanto dovrebbe esserci sia un’andata che un ritorno, la distanza risulta essere distanzaMax/2. [↑](#footnote-ref-5)
6. Retta passante per i due punti formati dall’intersezione di due circonferenze. [↑](#footnote-ref-6)
7. Vedere “Appendice 1” per le modifiche. [↑](#footnote-ref-7)