Team 6
Riccardo Castellani
Edoardo Mura
Andrea Pacilli

Mechatronics' project

Purposes

The goal of our project was to build an hunting and tracking system that is able to follow an object moving in the surrounding space. The system will use a Ping sensor to sense the moving objects, and will use two servo motors to rotate it on two different axes.
The Basic Stamp will command the whole system and an appropriate algorithm will drive the sensor through the object hunting.
The sensor will continuously rotate until an object falls in its range of activity. As soon as the object is hooked up the system will follow it. In case the Ping looses sight of the hunted object, the system will start again a sweep search in the space until the object is captured again.
The basic principles of the algorithm are the following:

- the system has to work in an open space and will use the ability of the Ping sensor to measures the distances to detect objects in an open space;
- when the object is captured the Ping will stop rotating and an LCD display will continuously update the position and the distance of the detected object;
- when a movement is detected the servo motors will be appropriately controlled to move the sensor and keep it in touch with the target.

A compass, along with servo rotation angles, will also be used to measures the target position in the 3D space.

Hardware solutions

The following hardware components have been used:

- Ping sensor
- Standard servo motor (2)
- Hitachi HM55B Compass Module
- LCD display
- 555 Timer (2)
- Piezo speaker
- Diodes (2)
- Capacitors & Resistors (several)

The Ping sensor

The Ping sensor is an ultrasonic sensor that is able to measures the distance of an object by computing the time that an ultrasonic wave takes to go back and forward. The Ping transmits an ultrasonic burst and returns an output pulse proportional to the time required for the burst echo to return to the sensor.

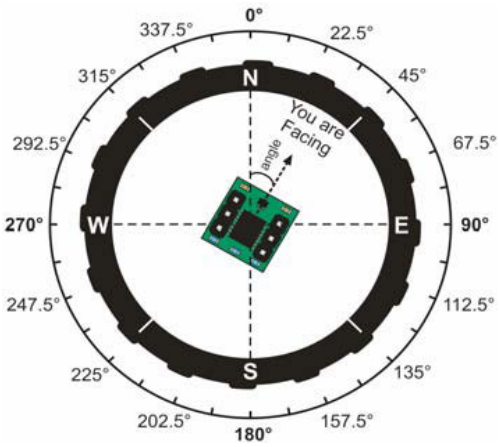
The Standard servo

The Standard servo motors allow movements in a range of 0 to 180 degrees, by the way, only a portion of the available range will be used. The first servo will support the Ping and will rotate on the vertical axis, while the second one will rotate on an horizontal axis and will be connected to the first servo in such a way to move both the Ping sensor and the other servo.

The vertical servo will range an angle of approximately 157.5° and will move left to right from the position 1000 to 250 and the other way round right to left from 250 to 1000.
The horizontal servo will move up and down from the position 1000 to 550 and backwards.

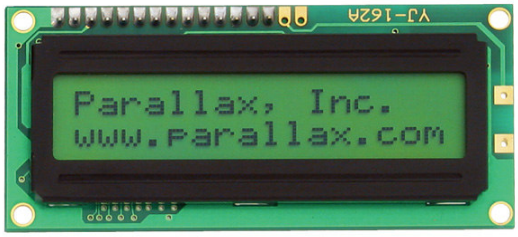
The compass

A compass module (Hitachi HM55B) will continuously monitor the direction of the Ping sensor, returning the angle between the detected object and the magnetic Nord.
By the way, because the compass has to stand on a plane and steady surface, an algorithm has been implemented to add to the magnetic angle detected by the compass to the angle ranged by the vertical servo. The compass will be standing on the steady support of the radar.



The LCD

A 2x16 serial LCD display will be used to report the activity of the Ping sensor. The display will output the position and the distance of the object, while if a pushbutton is pressed it will display the altitude.



The speaker

A sound emitter (piezo speaker) will be turned on to communicate the observers of an object detection.

The 555 timer

Two 555 timers, connected in cascade, will be used in the astable mode to drive the piezo speaker; such off-loaded configuration for the speaker beeping will prevent the BS2 from slowing down and focus its computational capabilities where truly needed. The purpose of the piezo speaker is beeping when the object is detected.
To achieve this goal, each of the two 555 timer has a specific task:

- First 555: when powered up, sends high-low pulses to the second 555 timer, setting the beeping rate;
  Duty cycle is set to 50% through a diode in parallel with R2; $R_1 = R_2 = R = 10k\Omega$, $C = 10\mu F$, hence resulting in:
  $t_{high} = t_{low} = t = 0.693 \times R \times C = 69.3ms$
  $F = \frac{1}{2t} = 7.215Hz$
- Second 555: driven on from the first one, generates a ~4.8kHz high-low pulse sequence (square wave) that induces the piezo crystal to vibrate. Such frequency is purposely close to the piezo resonating frequency to get a louder sound.
  Duty cycle is set to 50% through a diode in parallel with R2; $R_1 = R_2 = R = 10k\Omega$, $C = 0.015\mu F$, hence resulting in:
  $t_{high} = t_{low} = t = 0.693 \times R \times C = 0.1039ms$
  $F = \frac{1}{2t} = 4.81kHz$

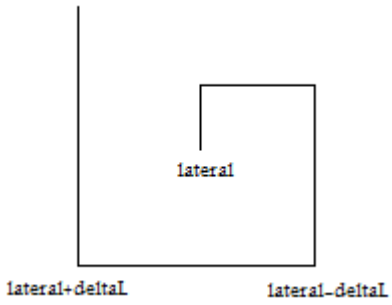
Algorithm and software solutions

The code structure

The algorithm is mainly divided in four pieces, one for each movement direction of the Ping sensor, left, up, right and down.
The Ping will start rotating by tracking a squared spiral around the centered position. As soon as an object is detected the Ping will stop and the LCD will display the corresponding information.
A main *Do loop* is used to track the spiral *until* the boundaries are not exceeded to avoid damaging the servo motors.
Four big *For next* loop are used to move the motors in the different directions, and in each loop the algorithm will be able to measure the distance of the object using the Ping, display the results of the hunt on the LCD and start a sub *Do loop* if the distance if smaller than a prefixed one.
Once we enter this loop it means an object has been found and the Ping will continue monitoring the situation, the compass will measure the position and the BS2 will display on the LCD screen the information we were looking for. The loop will stop as soon as the distance exceeds again the prefixed limit.
Because the Ping is tracking during the hunt a square spiral, that is basically a sequence of squares with bigger sides, the *For* loops that are used to control the servo motors have counters that are variables (*maxtimerA* and *maxtimerL*) that changes every time depending on the size of the square.
Similarly, because we had to move the motors in different positions every time, we had to monitor the duration of the pulse that was used on the *Pulsout* command for the servos. The duration is then a variable (for example *lateral+deltaL* for the left subroutine) that changes every time the main *Do loop* is crossed; depending on which direction the Ping is moving, a different *delta* is added or subtracted to the duration corresponding to the starting position (*lateral*).
Once we have a way of monitoring the motor's position going left to right, and up and down, we are able to compute the corresponding position in geometric degrees. In such a way we are able to relate this angle with the one given from the compass showing the magnetic Nord and we can the display on the LCD the position in the absolute frame of reference.
First of all the compass is fix on the rigid support and lined up with the frame of reference attached to the base. Then by adding the angle swept by the Ping projection in the compass plane to the one measured by the compass itself we are able to find the desired position.
Both angles of the rotating servos can be easily found using a proportion between the counter used for the servos and the physical angle of *158°* swept by the sensor (or *80°* depending on which servo we are looking at). In fact if the counter *timer* was growing from 1 to *maxtimer* and an object is detected, the *For loop* momentarily stops by entering the sub *Do loop*. Anyway the value of the counter *timer* is stored and can be used. For example if an object is detected when the counter is equal to *timer* we have that the angle swept by the Ping is

$$partialAngleL = ((2 * deltaL * timer) / \max timerL - deltaL) + lateral$$

because the sensor was moving from a position that was *lateral-deltaL* to a new position *lateral+deltaL* it was seeping a distance of *2deltaL*.
That is the distance we have to scale with the ratio *time/maxtimer*. Then at the ratio just found we must subtract *deltaL* and add *lateral* but, to avoid the risk of going negative, we will compute first the addition and then the subtraction.



$$partialAngleL = (2 * deltaL * timer) / \max timerL + lateral - deltaL$$

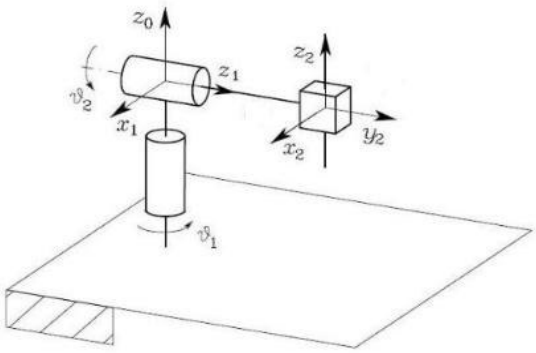
The result just obtained has to be now transformed in physical degrees, and this will be done using the following proportion

$$partialAngleL = (partialAngleL * 158) / 750 = partialAngle * *13806$$

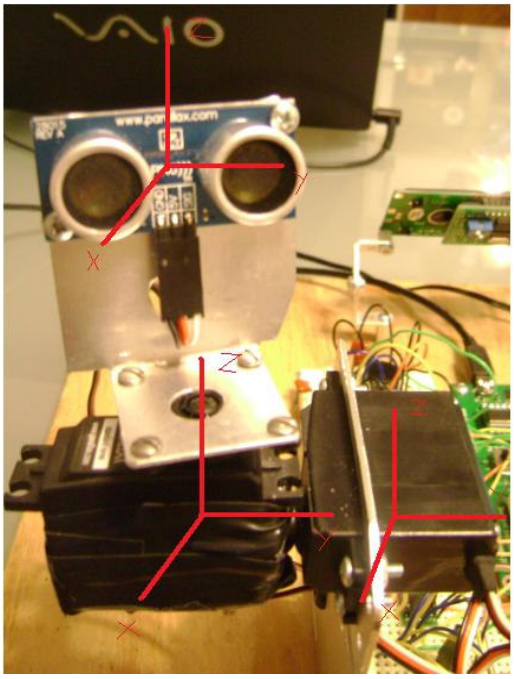
The math just shown is of course valid only for one of the four direction, different equations must be used in the other cases but the idea stays the same.

The rotation matrix

Our problem is cinematically similar to the one shown in the picture below.



So, the two angle ϑ_1 and ϑ_2 are the one swept by the servos, while we can see the little cube as the Ping sensor mount on the second servo. Our target will be moving on the frame of reference attached to the Ping sensor while the position outputted by the algorithm has to be the one relative to the frame of reference attached to the solid base.
To achieve such measures we had to use some mechanics basic theory and we had to compute the various rotation matrixes.
The frames of references used in our project are the one shown below



The frame attached on the Ping will be named *x''y''z''*, the one attached to vertical servo *x'y'z'*, while the one that is fix and attached to the base will be *xyz*.
The coordinates in the *x'y'z'* frame can be report in the fixed frame using the R^y matrix because the *x'y'z'* rotates on the *y* axis. Then we will have our coordinates as

$$x = R^y x'$$

and similarly we have for the other

$$x' = R^z x''$$

If we now rename φ and ϑ the two angles we have

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \cos\varphi & 0 & \sin\varphi \\ 0 & 1 & 0 \\ -\sin\varphi & 0 & \cos\varphi \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} + \begin{bmatrix} 0 \\ -l_1 \\ 0 \end{bmatrix}$$

and

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos\vartheta & -\sin\vartheta & 0 \\ \sin\vartheta & \cos\vartheta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x'' \\ y'' \\ z'' \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ l_2 \end{bmatrix} = \begin{bmatrix} \cos\vartheta & -\sin\vartheta & 0 \\ \sin\vartheta & \cos\vartheta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} d \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ l_2 \end{bmatrix}$$

Where l_1 is the distance between the frames *xyz* and *x'y'z'*, l_2 the distance between *x'y'z'* and *x''y''z''*, while *d* is the distance detected by the Ping. The distance *d* of course will be only on the *x''* coordinate due to the way the Ping was aligned with *x''y''z''*.
The final coordinates will then be

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} d \cdot \cos\vartheta \cdot \cos\varphi - l_2 \sin\varphi \\ d \cdot \sin\vartheta - l_1 \\ d \cdot \cos\vartheta \cdot \sin\varphi + l_2 \cos\varphi \end{bmatrix}$$

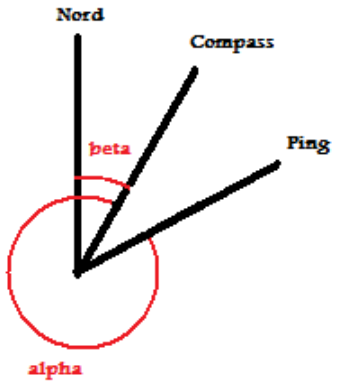
And those will be the one computed by the algorithm. Then to find the angle swept in the plane of the compass we will have to compute the tan^{-1} function of the coordinates

$$\alpha = \tan^{-1}\frac{y}{x}$$

This angle will then be added to the one measured by the compass using the following equation

$$TotalAngle = (360 - \alpha) + \beta$$

where β is the one measured by the compass.



Because the *partialAngles* computed in the previous paragraph are the one relative to the left and down positions of the servos to found the ϑ and φ angles used in the matrixes we

have to subtract *75°* and *40°*. During this calculus the angle could go negative and the BS2 would not notice it.
We then had to implement a little algorithm to monitor the signs of those angles and consequently adjust the equations in the rotation matrixes. The code can be found in the *Compute_Angle* subroutine in the main code attached at the end of the report.

The Ping sensor

To use the Ping sensor the commands *Pulsout* and *pulsing* have been use as shown below.

```
PULSOUT 13, 5
PULSIN 13, 1, time
cmDistance = 2260 ** time
DEBUG HOME, "Target distance:  ", DEC3 cmDistance, " cm "
```

The command *Pulsout* is used to generate a pulse on *Pin 13* with a width of duration of *5* in *2 μs* units.
The command *Pulsin* on the other hand is used to measure the width of the returning echo. The *Pin* used is always the same and is the one that controls the Ping sensor, *Pin 13*. The command *Pulsin* is able to measures the duration of a certain state (in our case 1 that corresponds to high) and stores the results in a variable (*time*).
Because the duration is measured in units of *2 μs* to find the distance in *cm* we have to compute the following passages:

$$dis = c_{air} \cdot t$$

Where *dis* is the distance, c_{air} the speed of sound in the air and *t* is the time stored by the command *Pulsin*.
Because the pulse goes back and forwards we have to divide *dis* by *2*

$$dis = \frac{c_{air} \cdot t}{2}$$

Then, because we want our measure in *cm* we will have to multiply everything by *100* and because the time is measured in BS2's units, of *2 μs*, we also have to multiply *t* by *2* and divide it by 10^6

$$dis = \frac{100 \cdot c_{air} \cdot t}{2} \frac{2}{10^6} = \frac{c_{air} \cdot t}{10^4}$$

Now we have to take into account the speed of sound in the air that is approximately *344.8 m/s* at room temperature

$$dis = 0.03448 \cdot t$$

The problem now is that the BS2 program can't handle numbers and multiplications for numbers less than unity. This is the reason why we have to use the operator ** *multiply high*, these operators multiplies two numbers returning the high 16 bits of the result.
So we now have

$$dis = (0.03448 \cdot 65536) **t$$

At this point we have stored in the variable *dis* the distance in *cm* and we can now work with it.

The Standard servo

The Standard servo is a servo motors that is able to rotate of a certain angle back and forwards. This micro controlled motor receive sequences of high and low signals that tells him how much and in which direction to rotate.
Normally to move such a servo a *For* loop would be used in the following way

```
FOR COUNTER = 1 TO 150
    PULSOUT 14, 750
    PAUSE 30
NEXT
```

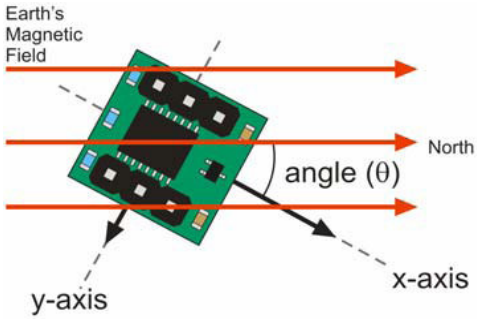
Such a loop delivers *150* pulses while the command *Pulsout* tells the BS2 where to send those pulses (*Pin 14* in this case) and their duration. In such a way we can control the position that will be reached by the servo, in fact the larger is the duration of those pulses the furtherer the servo will be moved. The *pause* is used to separate a pulse from the other.
Another very important parameter is the counter, if it is too small the servo will not reach the desired position while, if too long, there will be a pause before the next command is executed; this is the reason why the choice of this counter is fundamental for an optimization of the algorithm.
For our project two servos have been used. The first rotating on a vertical axis, from left to right, and the second up and down on an horizontal axis. The angle range was decide in such a way to not interfere with other hardware parts and to sweep a sufficient space.
The first servo, connected to *Pin 14*, is able to sweep approximately *160°* (that is in the pulse duration language from 250 to 1000), while the second one, controlled by *Pin 15*, will sweep more or less *90°* (from 550 to 1000).
The following commands were used to control our system

```
FOR timer = 1 TO maxtimer
    […]
    PULSOUT 14, lateral+deltaL
    PAUSE 30
    […]
NEXT
```

as we can see the duration of the pulse is a variable (*lateral+deltaL*) so the path swept by the servo changes depending on the side of the spiral that is sweeping. Consequently because the duration is a variable, also the *counter* has to change and goes from *1* to *maxtimer*, depends on the side.

The compass

The compass is able to return the *x* and *y* magnetic field strength measurement as shown in the picture below.



Of course if the *x* component is maximum and the *y* component is null, the compass is facing the magnetic nord otherwise the angle θ is given from the relation

$$\vartheta = \tan^{-1} \frac{-y}{x}$$

The following code was use to return the request variables

```
HIGH En: LOW En
SHIFTOUT DinDout,clk,MSBFIRST,[Reset\4]
HIGH En: LOW En
SHIFTOUT DinDout,clk,MSBFIRST,[Measure\4]
status = 0
DO
    HIGH En: LOW En
    SHIFTOUT DinDout,clk,MSBFIRST,[Report\4]
    SHIFTIN DinDout,clk,MSBPOST,[Status\4]
LOOP UNTIL status = Ready
SHIFTIN DinDout,clk,MSBPOST,[x\11,y\11]
HIGH En
IF (y.BIT10 = 1) THEN y = y | NegMask
IF (x.BIT10 = 1) THEN x = x | NegMask
```

High En: Low En and the first *Shiftout* command are used to reset the compass. The second sequence of *Highs* and *Lows* with the second *Shiftout* are used to start the measurement and *status=0* is for clearing the previous status.
In the *Do* loop first a measurement of the status command is sent to the compass using the *Shiftout* while later, a *Shiftin* command is used to acquire the status. The *Do* loop will then be stopped when the condition *status=Ready* is satisfied.
With the next *Shiftin* the value of *x* and *y* are returned and with the last *High* command the module is turned off. The last two *If* conditions are used to deal with negative numbers.
In such a way we are able to compute the angle between the frame of reference used in the algorithm, that is the same of the compass, and the magnetic Nord. Then by adding and subtracting this angle to the one swept by the Ping we can monitor the position of our object in the space.

Conclusions and further developments

The principals limits in our project where given form the technical specifics of the Ping sensor. First of all the range limits within which a target can be detected goes from 2cm to 3m. Then, the width of the "visible" cone doesn't permit an accurate directionality.
At the same time the velocity of the servos doesn't permit a fast search of the lost target.
Then to improve the performances of the system a more accurate sensor and faster servos would be enough.
Anyway, just by increasing the velocity, the performances may not increase, in fact the high inertias play a fundamental role. A delay in the response was noticed when an object is detected and the command to stop the servo is given.