DESIGN OF AN INCREMENTAL ENCODER

Cazenave Thibaut
Costa Nicolas
El Yaalaoui Mourad
Quéré Mikael
The aim of this report is to design an incremental encoder needed in order to compute the position and the speed of the talrik mobile robot shown in the figure below (picture by Nicolas Costa):

1. The implantation of the encoder

The basic principle of the encoder is to stick a black/white band on a wheel, send light on it and detect reflection. This principle is shown in the figure below (drawing by http://manubatbat.free.fr):
So we looked for an adapted component and found the EE-SY410. It’s made of an emitter (Led) that emits light and a receiver which creates a low level output voltage when light is on and a high level output voltage when light is off. A limiting condition to a good performance of the receiver is the sensing distance which shouldn’t be more than 5mm. To buy the component, look at [www.radiospares.fr](http://www.radiospares.fr) and then “composants d’automatisme, capteurs de proximité et cellules photoélectriques, miniatures”. It is then referred as “détecteur photoélectrique à réflexion diffuse série EE-S”. The documentation of the component can be found at [http://oeiweb.omron.com/oei/PDF/D21SY310410313199.pdf](http://oeiweb.omron.com/oei/PDF/D21SY310410313199.pdf).

The price is 6.97€ if you buy from 1 to 9 components and 6.62€ if you buy 10 or more components. The wheel where we plan to place the system is represented in the figure below (picture by Olivier Dechazal):

We place the photomicrosensor EE-SY 410 at a sensing distance of 5 mm from the wheels thanks to a plastic box. We actually do not found any appropriate box, so we plan to build it according to the following scheme:
NB: In the scheme above, we do not draw the holes which are necessary for the 5 pins of the photomicrosensor (2 inputs and 3 outputs). Indeed, we plan to make one fence in the box for the connector linked to the pins of the EE-SY410 photomicrosensor.

The dimensions have been calculated such that the photomicrosensor is located at sensing distance of 5 mm from the wheels and such that the box does not touch the ground. The box must be fixed to the robot thanks to 2 screws. We need two 3 pin-connectors (outputs), two 2 pin-connectors and two photomicrosensors (one for each wheel). The outputs of the photomicrosensors are: the supply voltage Vcc=5V, the ground, and the output. We connect through a 3-pin connector (available in the laboratory) the photomicrosensor to the microcontroller by using pins IRDT13 & 14. A 2-pin connector is used for the supply of each of the two photomicrosensors. The linking scheme of the component is represented on its data-sheet.

Sensing object: We use a self-adhesive black/white band with a reflection factor between 15% and 90%. The diameter of the wheel will be about 57 mm and the width between two holes will be about 10.4 mm.

The last remaining work is to fix the boxes containing the two photomicrosensors and to connect the pins. We plan to insert in each box a little electronic card manufacturable in any lab, the aim of this card is to
support the photo sensors and the connectors. Also, we might need some solders.

2. Linkage between the encoder and the computer

It shouldn’t be difficult to interface the encoder with the robot, as its output level is dependent on the Vcc value which goes from 4.5 to 16V. One way would be to use one of the 2 digital inputs of the sensor expansion board (MRSX01) which are mentioned in the Talrik documentation page 48 but on which we couldn’t find much information. The other way would be to use the IRDLW and IRDRW detectors’ connectors. These detectors are optional so it’s necessary to check if they are actually used in our robot. The connectors associated with them (IRDT13 & 14) lead to an A/D converter which would force us to implement a pass-thru TTL gate to ensure that the level is sufficient to make the converter see only ‘1’ and ‘0’. The connectors are 3-pin male headers which is an advantage: it supplies 5V and ground. Moreover it looks quite simple to activate a connection as mentioned in the Talrik documentation; we just need to insert this line in the C-program of the robot:

```c
mux_sel(IRDT13) = mux_sel(0x08) ;
```

And then the values stored into AtoD[0] every 2 msec. will come from the detector.

So this solution is better, but again we need to check if IRDT13 & 14 aren’t already used.

Now that the binary information has reached the microcontroller, it’s necessary to calculate the frequency of ‘0’ and ‘1’. To do that the program of the robot has to count the number of changes between ‘0’ and ‘1’ during a loop fixed on a high frequency constant clock supplied by the microcontroller.

3. Determining the position and the speed of the robot :

The computer has now an information about the frequency of the input signal. We’ll see how, with this frequency, the system can computes the speed and the relative position of the mobile robot.

It’s important to disclaim that all our theory is based on the assumption that there is no drift of the wheels.

3.1. Computing of the speed of each wheel :
Let $f$ be the frequency of the input signal, and $d$ the length of a tooth in cm. We can determine the distance runned in one second, it is equal to:

$$\delta = 2.d.f$$

Because the speed in cm/s is equal to the distance runned in one second, we can compute $s_1$ and $s_2$ the speed respectively of the wheel 1 and the wheel 2. (Remember that, in our case, $d = 5.6$ mm).

More over, the speed of the mobile robot will be assumed as its linear speed. Then we can compute the speed of the robot as the lower speed of its two wheels.

3.2. Computing of the position:

First of all, we will consider a reactualization of the system each second. It’s far away from the potential computing speed of the system but it simplifies the computings and allows an easier debugging during the development step. But if we want a faster system, we should divide the quantities $s_1$ and $s_2$ as we want to reduce the sampling time, i.e. if we want a half sampling time, we should divide by 2 the sampling time.

In a second time, we will define the parameters of the mobile robot. There are 2 parameters:

- The position of the mobile (coordinates in the 0xy plane)
- The orientation of the mobile (vector $o$)

The parameters of the mobiles are shown in the figure below:
Length between the 2 wheels

We can write the 2 parameters of the mobile like this:

Position:

\[ P[t] = (x[t], y[t]) \]
\( x \) is the x-coordinate of the mobile
\( y \) is the y-coordinate of the mobile

Orientation:

\[ o[t] = o1[t] \cdot i + o2[t] \cdot j \]
\( o \) is the first coordinate of the vector
\( o2 \) is the second coordinate of the vector
\( i \) and \( j \) are reference vectors of the plane
\( i \ (1,0) \) & \( j \ (0,1) \)

\( t \) is an integer number which corresponds to the sampling time, that's why we work with discrete data.

### 3.2.1 Initialization process:

\( o[0] = (0,1) \) & \( P[0] = (0,0) \)

### 3.2.2 Evolution equations:

We must divide the process in different cases, because the movement will be linear if \( s1 \) and \( s2 \) are equal and will have a circular component if \( s1 \) and \( s2 \) are not equal.

#### 3.2.2.1 \( s1=s2 \):

\[ P[t+1] = (x[t+1], y[t+1]) \]
\( x[t+1] = x[t] + o1[t] \cdot s1 \)
\( y[t+1] = y[t] + o2[t] \cdot s2 \)

\( o[t+1] = o[t] \)
\( ( \text{if the movement is linear, the orientation is invariant}) \)

#### 3.2.2.2 \( s1>s2 \):

Let \( \alpha = (s1-s2)/l, \)
The computing of \( P[t+1] \) is more complex, we have to divide the movement in two parts: a linear component and a circular component.

\[
\begin{align*}
  x[t+1] &= x_1[t+1] + x_2[t+1] \\
  y[t+1] &= y_1[t+1] + y_2[t+1]
\end{align*}
\]

where \((x_1,y_1)\) is the linear component and \((x_2,y_2)\) is the circular component.

Then, we can write for the linear component:

\[
\begin{align*}
  x_1[t+1] &= x[t] + o_1[t].s_2 \\
  y_1[t+1] &= y[t] + o_2[t].s_2
\end{align*}
\]

and for the circular component:

\[
\begin{align*}
  x_2[t+1] &= x[t] + l.( -o_2[t] + \cos(\alpha).(-o_2[t])) \\
  y_2[t+1] &= y[t] + l.( o_1[t] + \sin(\alpha).(o_1[t]))
\end{align*}
\]

it gives us:

\[
\begin{align*}
  x[t+1] &= x[t] + o_1[t].s_2 + l.(-o_2[t]).(1+\cos(\alpha)) \\
  y[t+1] &= y[t] + o_2[t].s_2 + l.(o_1[t]).(1+\sin(\alpha))
\end{align*}
\]

The new orientation vector is given by:

\[
\begin{align*}
  o[t+1] &= o_1[t+1] * i + o_2[t+1] * j
\end{align*}
\]

with \( o_1[t+1] = \cos(\alpha).o_1[t] \)
\( o_2[t+1] = \sin(\alpha).o_2[t] \)

### 3.2.2.3 \( s_2 > s_1 \):

There are a few modifications with the previous case

Let \( \beta = (s_2-s_1)/l, \)

\[
\begin{align*}
  x[t+1] &= x[t] + o_1[t].s_1 + l.(o_2[t]).(1+\cos(\beta)) \\
  y[t+1] &= y[t] + o_2[t].s_1 + l.(-o_1[t]).(1+\sin(\beta))
\end{align*}
\]

and

\[
\begin{align*}
  o_1[t+1] &= \cos(\beta).o_1[t]
\end{align*}
\]
\[ o_{2[t+1]} = \sin(\beta) . o_{2[t]} \]

3.3 General Algorithm:

We are now able to write a general algorithm which resume all the cases.
The system is now designed and we can go to the manufacturing step, this step can be easily done with some lab-tools. Moreover we can expect an evolution of the system with a control of the position.