Project 1
Report

January 2003
1. Introduction

The robot studied is Talrik\textsuperscript{ITM}. It is an autonomous mobile robot with a circular shape. It rides on two wheels and a rear caster:

This robot is programmable, which is going to allow us to manage its velocity and its position. So we first need to know these two characteristics. A good solution can be designing differential encoders. Actually, these encoders would be placed on the wheels and used to do these measures.

First, two classical solutions are proposed: the segmented disk and the one based on code wheel. After having described these two techniques, the solution chosen has been developed as for its technical criteria and its cost. Finally, the connections and an algorithm for determining the position and the velocity from the encoder measurements have been planned.
2. Description of basic principles of rotary encoders

Encoders can be categorized into photoelectric (or optical), mechanical contact, and magnetic types. Photoelectric encoders – due to their high accuracy, high reliability and relatively low cost – play a significant role in machine tool technology. Most of today's rotary encoders operate on the principle of the photoelectrical scanning of very fine gratings.

As reported on the mmsonline website: “The scanning unit in an encoder consists of a light source, a condenser lens for collimating the light beam, the scanning reticle with the index gratings, and silicon photovoltaic cells. When the scale is moved relative to the scanning unit, the lines of the scale coincide alternately with the lines or spaces in the index grating. The periodic fluctuation of light intensity is converted by photovoltaic cells into electrical signals. These signals result form the averaging of a large number of lines. The output signals are two sinusoidal signals that are then interpolated or digitized as necessary.”

Rotary Encoders

Rotary encoders are used in measuring rotational movements but they are also often used in measuring linear movements.

Incremental Rotary Encoders

The output signals of incremental rotary encoders are evaluated by an electronic counter in which the measured value is determined by counting "increments". These increments are independent from the rotation direction. The disk is made by plastic and it has holes or, more commonly, there are some transparent areas, where passes a light beam.
Quadrature Encoders

On the toplevelserver.mi.infn.it web site is explained: “The most common type of incremental encoder uses two output channels (A and B) to sense position. Using two code tracks with sectors positioned 90° out of phase (figure below), the two output channels of the quadrature encoder indicate both position and direction of rotation. If A leads B, for example, the disk is rotating in a clockwise direction. If B leads A, then the disk is rotating in a counter-clockwise direction. Therefore, by monitoring both the number of pulses and the relative phase of signals A and B, you can obtain both the position and direction of rotation.”

In addition, some quadrature detectors include a third output channel, called a zero or reference signal, which supplies a single pulse per revolution. This single pulse can be used for precise determination of a reference position.

Absolute Rotary Encoders

Absolute rotary encoders provide an angular position value, which is derived from the pattern of the coded disc. The code signal is processed within a computer or in a numerical control. After system switch-on, such as following a power interruption, the position value is immediately available. Since these encoder types require more sophisticated optics and electronics than incremental versions, a higher price is normally to be expected. The most commonly used coder is Gray coder, a unit-distance coder where only one single bit changes with the transition from one measuring step to the next. Natural binary coder is also frequently used for very high resolutions. With this code, more than one signal may change when going from one measuring step to the next.

January 2003
On the left: incremental encoder disk. On the right: 10 bits (1024 positions) absolute encoder disk.
3. Solution proposed

The two previous solutions tend to make us prefer an incremental encoder.

Among the different ones consulted on different catalogs, the first option was to design this kind of encoder thanks to a disk of stripped paper and one phototransistor. Even if it allows a low cost, the designing was not so sure to succeed since it generated a lot of “do it yourself”

That is why an optical incremental encoder was preferred. Nevertheless, its dimensions risked not corresponding to the ones of the robot.

Nevertheless, a kit that works on the same way as the previous encoder was found. Actually, it is composed of an optical encoder module and a disk.

Concerning the disk, many diameters were proposed but the 1" Disk from US Digital™ was chosen. It allows coding the wheels as explained in the previous part. To do it, an optical encoder is needed.

For more details about the disk, please look at the data sheet on the following web site: http://www.usdigital.com/products/disklinear

Concerning the optical encoder module, it is the model HEDS from US Digital™.

January 2003
As described on the data sheet of the designer, “this incremental optical encoder module is used to detect a linear or rotary position when used together with a codewheel. Each module consists of a lensed LED source and a monolithic detector IC enclosed in a small plastic package. They are available in 2 channel or 3 channel versions. The resolution and index version of the modules and codewheels must match. They can easily be mounted by using 4-40 screws through the mounting holes. These devices are very reliable when connected properly. Improper connections are the most common cause of failure”.

For more details about the optical encoder module, please look at the attached data sheet on the following web site: http://www.usdigital.com/pdf/data-sheets.shtml

January 2003
The important parameters are the dimensions of both the disk and the encoder module as explained below:
The last problem consists on determining where putting this encoder. In fact, it is possible on the wheels with the modules between two IR LEDs or on the wheels with the modules above the circular platform. A good solution seems to put the module above the circular platform as shown below. Actually, the module is fixed to the disk, which is on the wheel.

January 2003
Concerning the parts list, we can summarize the solution chosen in the following table:

<table>
<thead>
<tr>
<th>Designation</th>
<th>Quantity</th>
<th>Unity Price</th>
<th>Where to buy</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISK/LIN 1” disk &amp; linear strips</td>
<td>2</td>
<td>$7</td>
<td>US Digital: <a href="http://www.usdigital.com">http://www.usdigital.com</a></td>
</tr>
</tbody>
</table>

So the final cost is 64 $
4. Connection and algorithms

The goal of this part of the report is to show how to connect the sensors that will measure speed to the microprocessor, and to tell us about the constraints, limits…

Let’s begin with a small summary about the Talrik robot and its microprocessor.

The Talrik robot

It is based on a MRC11Motorola micro-processor. It also has the onboard sensor expansion board MRSX01 that extends the sensor capacities of the robot. And yet, it is not enough, because only 2 digital inputs are available, and we need four of them (two for each optical encoder). As a consequence, we will have to use analog inputs, of which we have 6, and apply certain rules to “digitalize” the values that we will measure, i.e. transform the continuous values in either 0 (if value is less than x volts) or 1 (if value measured is more than x volts).

We will use the ANALOG header to connect our sensors to analog inputs. To select, at any time, these inputs, we will only need to move the MMR register so that the SELx bits select the right input and feeds it (through the 3 multiplexers) to the PE1 input. This will be used through the 4AtoD[ ] data structure, that contains PE0…PE7 bits, and is loaded automatically every 2 milliseconds.

To change the value of the MMR register and choose a given input, we will use the following instruction:

\[
\text{mux\_sel(given input) = mux\_sel(0x??)}
\]

that will load the right bit pattern into the SEL field of the MMR register.

Analog inputs
Of course, we will need to amplify the level of signal from the sensors, because it will be far too low. For that, we can use any type of amplificatory (for example the 741 op-amp, it is cheap but non-linear. Anyway, this is absolutely not a problem for the kind of use we are planning) to bring levels to 5V.

It would be much easier if we could use digital inputs, if there were enough of them…

**Algorithms**

We have at least two ways of writing our algorithms. Directly in assembly code, which is very difficult (even when not working on SIMD machines…) or using Interactive C, and it will certainly be easier because IC is a higher-level language.

Anyway, at the moment we are just trying to find the algorithms to allow the robot to know how far it has gone, and what speed its wheels are turning.

Let’s see once more the nature of the signal we will use: let’s focus on the left wheel: its optical encoder sends two signals that will be “digitalized” by our algorithm. Once this is done, the signals A and B look like that:

<table>
<thead>
<tr>
<th>Phase shift</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

The first thing to do will be to determine the direction of movement, using the phase shift. After that, we can use only one of the two signals: the important information is each change in the value of the signal.

So we will suppose that the sensors changes value each time a wheel has turned of x degrees. We will call L and R the captors of the Right and Left wheels. We will store the number of “changes” of each captor in the variables r_count and l_count (or know if the values have changed, we store a previous value in l_value and r_value). Every once in a while (the time constant is to be determined) we check the values of r_count and l_count, and one is smaller we change the command of the motors to adjust the difference. We then reset the values of the two variables.
Here is the algorithm in natural language:

/*Initialisation */

l_count = r_count = 0 (no move has yet been made on this round)
l_value = r_value = 0  (0 is for a black sector, 1 for a white one)

/*Loop */

/*Begin of loop (loops to do during 200 ms) */

Select left sensor
if (left sensor value != l_value)
then increase l_count && change l_value

Select right sensor
if (right sensor value != r_value)
then increase r_count && change r_value

/*End of loop */

difference = l_count – r_count
If (the robot is supposed to go straight)
Then
    if (difference < 0)
    then increase the average power of left motor or decrease average power of right motor
    if (difference > 0)
    then increase the average power of right motor or decrease average power of left motor
Else do nothing

End

To that algorithm, we can add two variables that will store an “instantaneous” speed, based on the last iterations of the loop (so maybe the average speed in 200 ms). The calculus will be :

Speed of left wheel = l_count * (x degrees) / (360 degrees) * 2 * Pi * (radius of wheel)

January 2003
5. Conclusion:

The project has shown the very first steps of designing and finding implementation for a given robotic problem. Thanks to documentation and meetings together with the members of the team, a solution has been progressively established. Should it be the chosen solution for the two robots, the present report should be successful in leading the building of the encoders.

For further of information on the elaboration of this report, please refer to the following websites:

http://www.usdigital.com
http://www.minimotor.ch/minicatalog/pdf/Encoders/eE_HEDS.pdf
http://www.seattlerobotics.org/encoder/200010/dead_reckoning_article.html
http://www.galileimirandola.it/lezioni/automazione/lezione1/encoder.htm
http://topserver.mi.infn.it/mies/labview/instrupedia/instrupedia/appnotes/73417786256869005E5FC3.htm
http://www.mmsonline.com/articles/049703.html

January 2003