Line Follower Robot

Giovanni Accongiagioco
Simone Brienza
Daniele Giannetti

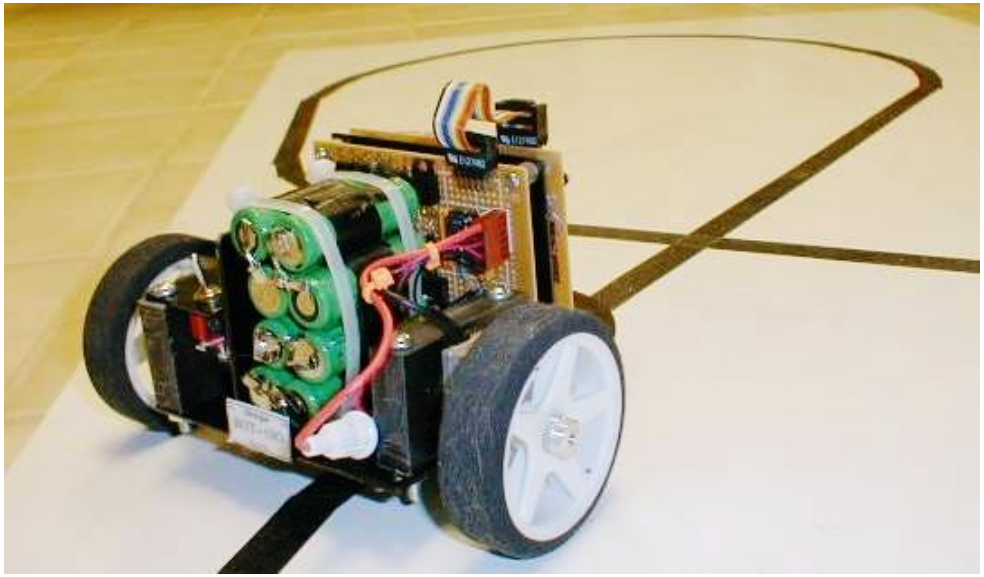
Introduction

Line Follower Robot

A line follower robot *(LFR)* is a simple type of robot whose only task is to follow a line on a differently colored plane.

Several examples of line follower robots are available on the Internet. Different technologies and solutions may be used to build these robots.

An example of line follower robot is given below.

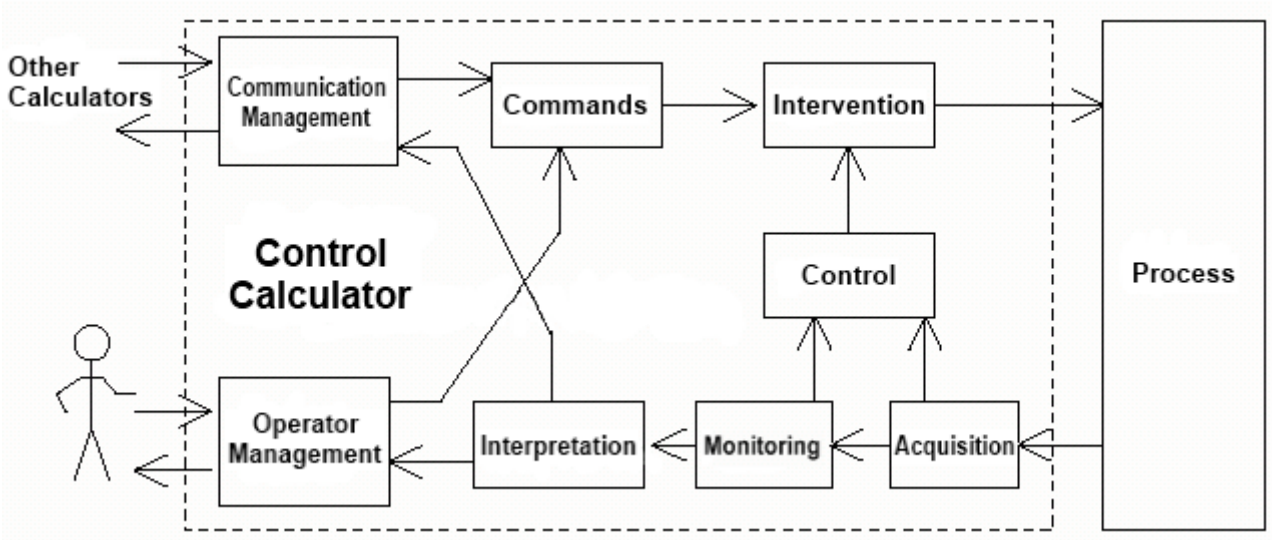


Introduction

Comparison with a generic Control Calculator

Even if realizing a line follower robot is not an example of classical industrial control, several technologies and problems faced are similar to those encountered in industrial control applications.

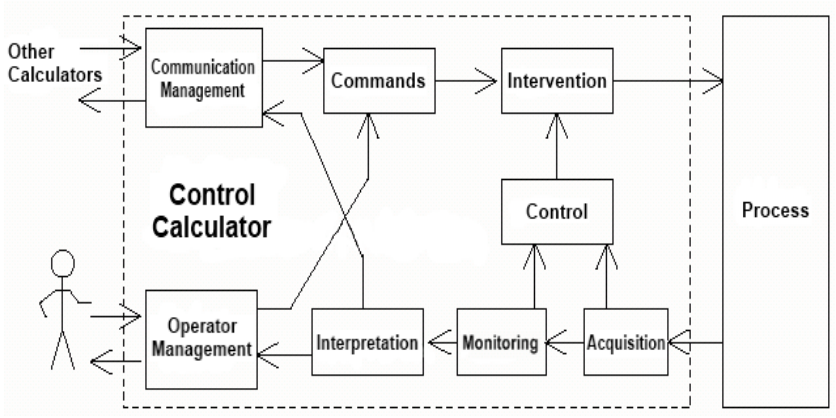
We may indeed compare the robot control system with a generic industrial control calculator. The functions of an industrial control calculator may be represented as follows:



Introduction

Comparison with a generic Control Calculator

Some of the typical functions of a control calculator are neither present nor requested when realizing a Line Follower Robot (LFR).
We may arrange them in the following list:



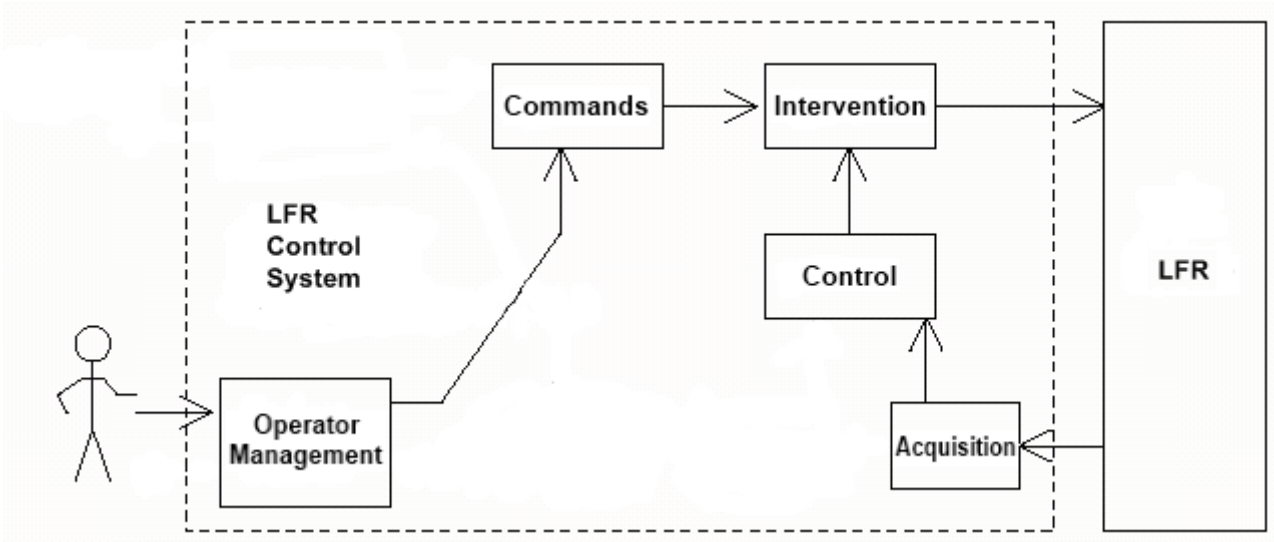
- *Communication Management* – we do not need to communicate with other control systems.
- *Monitoring* – we do not need a monitoring function because no synthesis or filtering of sensor readings is sent to other control systems or presented to the operator (except for debug mode).
- *Interpretation* – in absence of a monitoring function, the interpretation function is useless (the only role of this function is to transform the monitoring data in a useable form).

Introduction

Comparison with a generic Control Calculator

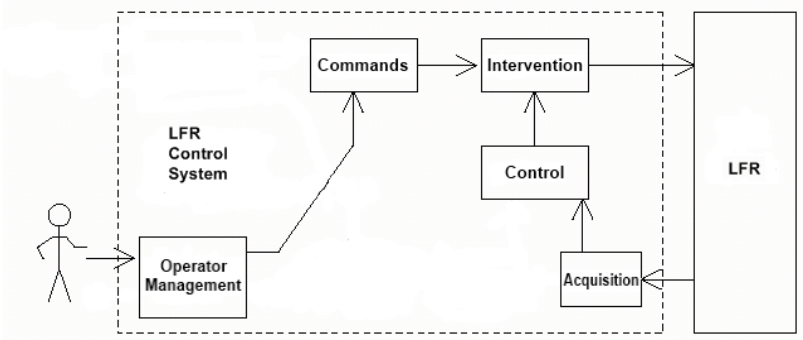
In our case we don't have a true interface with the operator, indeed there isn't a true operator. In our LFR we allow the user to send only particular commands to the LFR (a stop command and a reset command).

The LFR control system may be represented as follows:



Introduction

Comparison with a generic Control Calculator



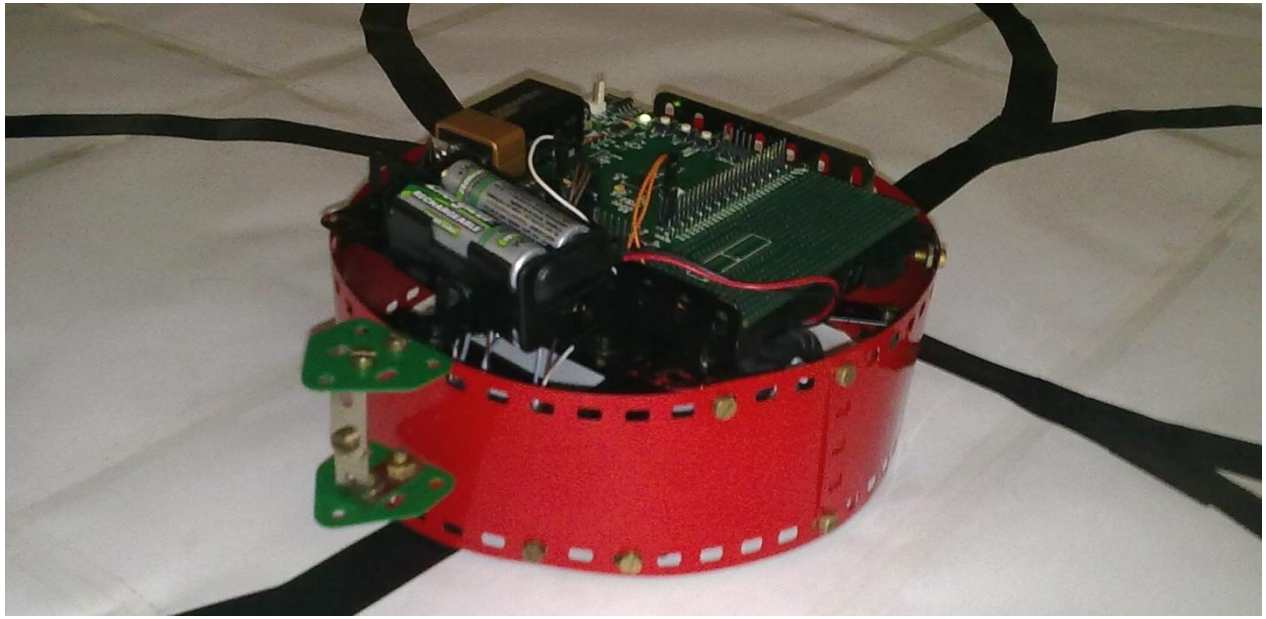
Where we have that:

- *Operator Management* – user interaction function, in our case we do not have a software task taking care of the user, just a physical interface through physical buttons on the board.
- *Commands* – the commands we have in the system are stop and reset, user actions are translated in commands through the ADuC836 interrupt system and do not need a particular task.
- *Acquisition and Control* – those two functions are realized in our case using a single task because the control function may be executed by using only a single reading (once for every sensor reading).
- *Intervention* – this function is realized in our case with a task that takes care of acting on the actuators, driving the robot where the control function decided the LFR should move.

LFR Hardware Structure

Generic Description

A picture of our line follower robot is given below:



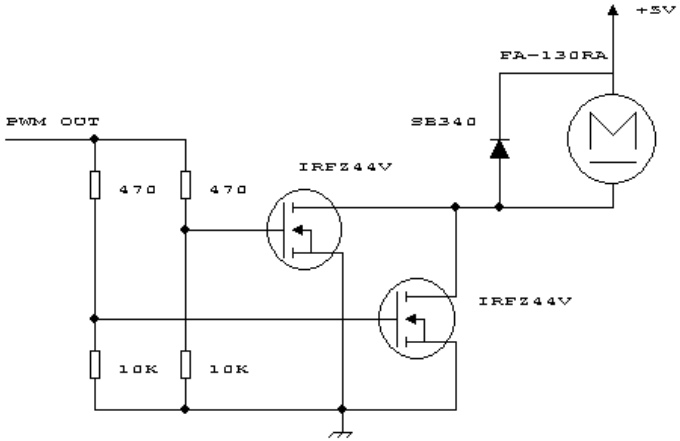
- AD MicroConverter Evaluation Board on the top floor
- Stripboard with motors and sensors driver circuits in the middle
- Two independent DC motors and wheels below the bottom floor
- Three IR reflective sensors below the bottom floor
- Battery pack to feed the motors, the sensors and driving circuitry on the bottom floor
- Ball caster as third contact point with the ground below the bottom floor

LFR Hardware Structure

Independent DC Motors

The *Tamiya 70168 double gearbox* was used to allow independent driving of two motors with configurable gear ratio.

The driving electronic circuit for each motor is built using a couple of power transistors and a safety diode as in the following picture.



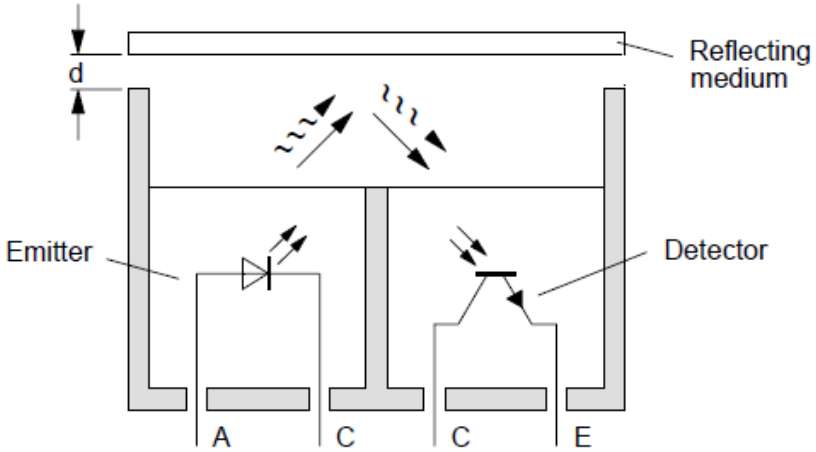
The motors are *Mabushi FA-130RA* motors with a maximum drained current of some A in stall conditions. This circuit may even be used with bigger motors if more power is needed.

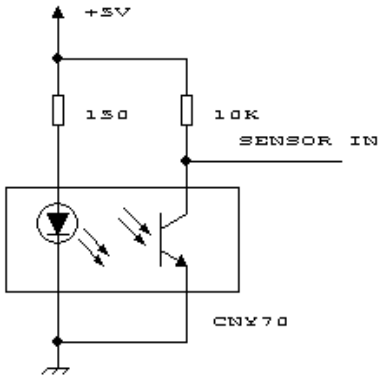
The driving PWM frequency is 655 Hz, based on commonly used frequencies for these low-voltage motors.

LFR Hardware Structure

IR Reflective Sensors

Three *Vishay CNY70* IR reflective sensors were used to detect the black track on white background. Each sensor is provided with an IR emitter and a detector as in the picture on the right.





The driving electronic circuit for each sensor is just a couple of resistors, as depicted on the left.

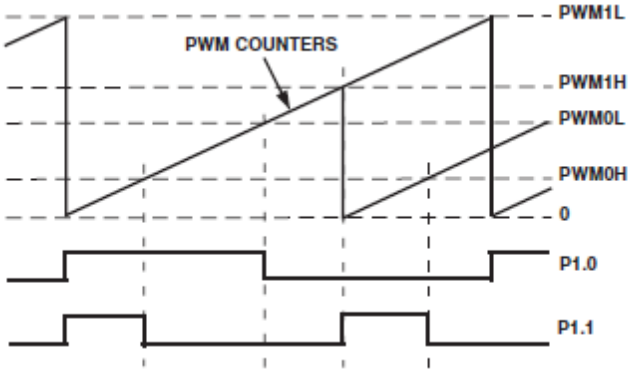
The three sensors are aligned orthogonally to the track, the reflectance of the black track is less than 10% while the reflectance of white paper is above 90%. This condition allows to use digital input pins directly to perform sensor readings.

LFR Hardware Structure

ADuC836 Peripherals – PWM

The built-in PWM generator of the ADuC836 microcontroller is used in mode 5 to obtain two independent PWM signals. The PWM thresholds and counter are configured to obtain two in-phase 655 Hz PWM signals with independent configurable duty cycle.

In mode 5 the PWM behaves as depicted on the right, we use the PWM1L = PWM1H condition to obtain the same frequency on both PWM signals.



Other PWM modes may be used in order to obtain independent PWM signals and configurable duty cycle with the desired frequency (e.g. mode 2). Mode 5 was chosen because it is the most flexible (allows independent configuration even of the frequencies, not needed now but maybe for future extensions).

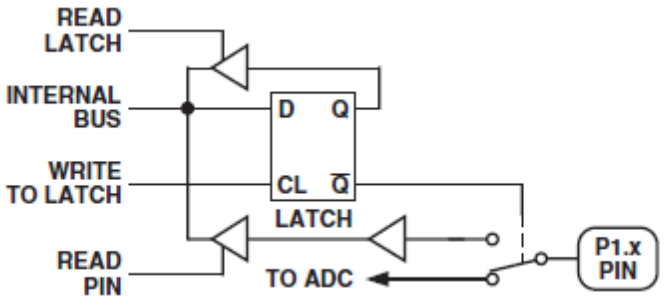
LFR Hardware Structure

ADuC836 Peripherals – Input pins

Three digital input pins are used for our IR reflective sensors.
Given the structure of the sensor driving circuit, we need to choose a port without internal pull-up resistors to avoid unexpected currents in the sensor circuit.

Pins 4, 5 and 6 of port P1 were selected. This is the preferred choice because:

- These pins are designed to be used only as inputs.
- If needed, these pins can be easily configured as analog inputs.



Pins 4, 5 and 6 were selected among all port P1 pins because other pins in the port having the above structure also have some different special function not needed for our purpose.

LFR Hardware Structure

ADuC836 Peripherals – More Peripherals

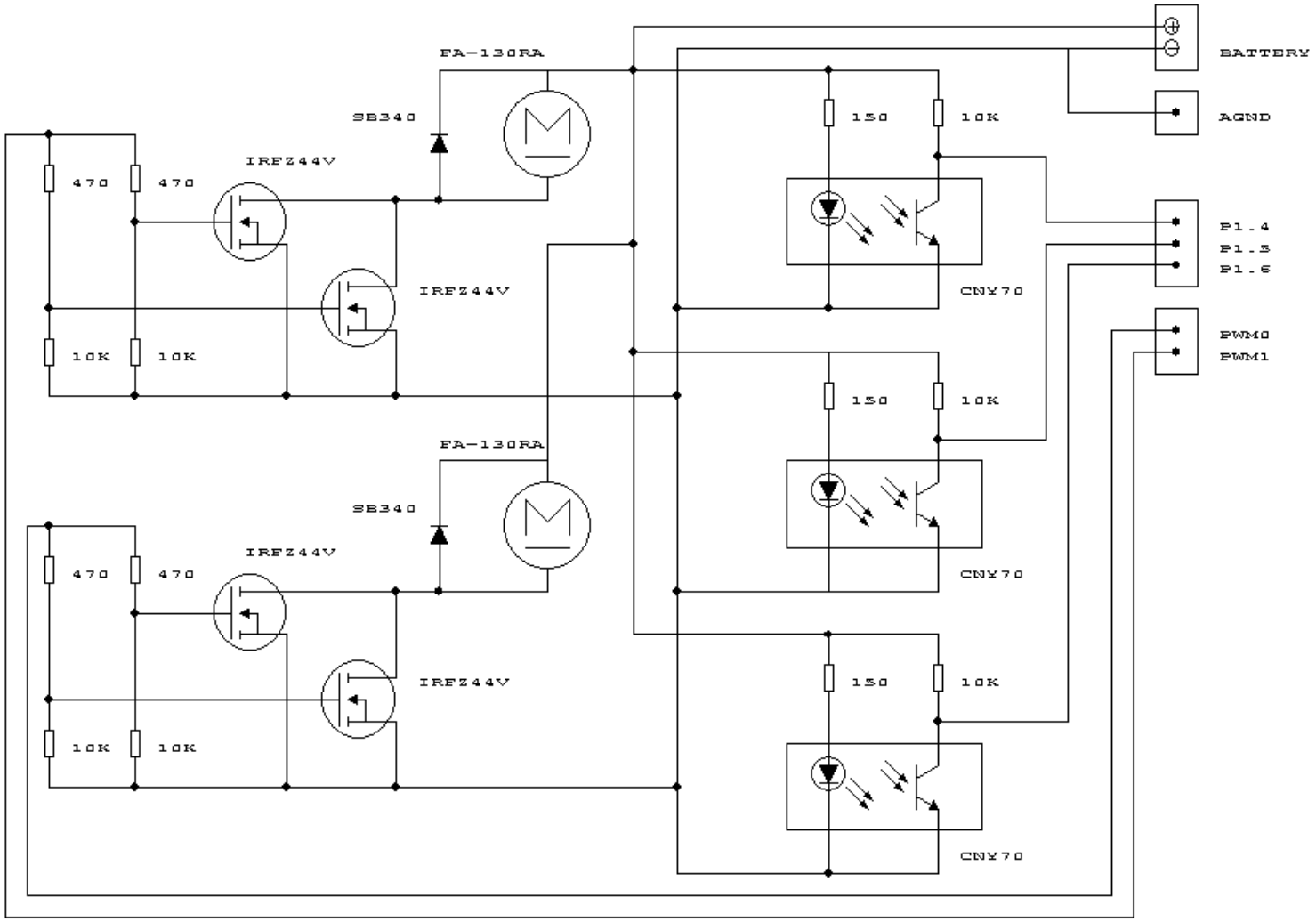
The built-in ***UART serial interface*** is used when the software is compiled in debug mode, in such case the serial interface is used to send to a PC listener some debug messages. Serial operations may interfere with the scheduler operations (see below); therefore, we recommend using the serial interface only for debug purpose. Timer 3 is used for serial operations if debug mode is requested.

Timer/Counter 2 is used for scheduler timing and is not available to the programmer. A scheduler is used to guarantee some soft real-time execution constraints and only works as intended if the programmer is not using the interrupt system.

LFR Hardware Structure

Full Circuit

In the picture below, the full circuit built for the LFR is depicted.



LFR Software Architecture

Scheduler

The scheduler from the book "*Patterns for Time-Triggered Embedded Systems*" by Michael J. Pont is used. Two tasks are present in the system during normal operation, a third in case of debug mode.

- **Task1**: Sensor reading and control commands computation
- **Task2**: Actuation based on the control commands

In case of debug:

- **Task3**: Debug messages writing to the serial interface

In the basic version of the scheduler, timer 2 is used to receive the scheduling tick every millisecond. Idle mode is used to save power until the next scheduling tick is received.

Using a maximum theoretical speed of 0.5 m/s, we want to sample (and actuate) every 20 ms to obtain at least one sample every cm of travel distance.
Debug messages (when requested) will be printed once every 0.5 s.

LFR Software Architecture

Scheduler

Therefore, using a scheduling tick period of 1 ms is a tremendous waste of computational resources and battery power. The scheduler has been modified to use a scheduling tick period of *20 ms* (and the two main tasks will be executed once every tick).

When using the debug mode, the behavior may become unpredictable because of the overhead introduced in the system: time constraints are not guaranteed to be satisfied.

External interrupt 0 is used in our case to receive the user command to stop the robot. In this case the interrupt is not dangerous for the scheduler activities because when this interrupt is received the motors are halted and the ADuC836 chip is powered down.

LFR Software Architecture

Line Following Algorithm

Task 1 allows to read the pins connected with the sensors. So, after inverting the whole P1 port, we have 3 bits containing the obtained values, which we call S2, S1 and S0.

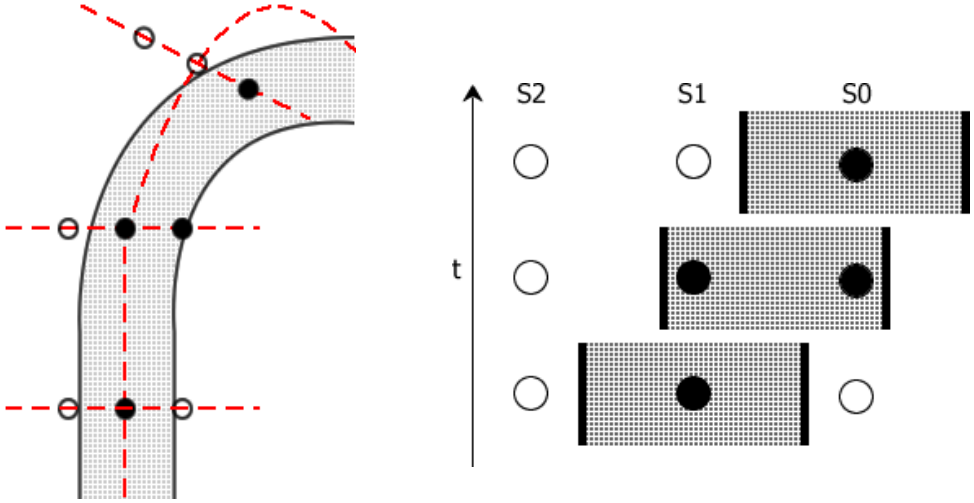
Knowing these bits, we are able to determine the movement the robot has to do:

- if only the central sensor returns 0 (sees black) then we are in the center of the line and we go straight on;
- if two near sensors (center and left, or center and right) return 0, then we steer slowly towards those sensors;
- if only a lateral sensor sees black we have to steer quickly instead.

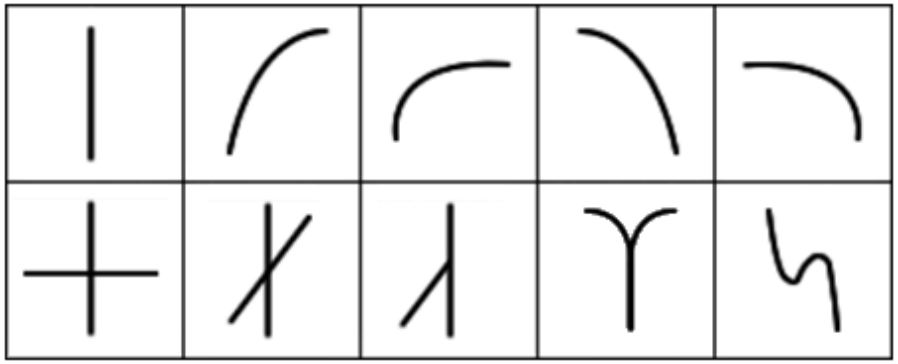
Through this algorithm we can react to different bends, turning more when the robot detects it's getting out of the circuit.

LFR Software Architecture

Line Following Algorithm



Example of behavior with a right bend and list of valid circuital elements.



LFR Software Architecture

Line Following Algorithm

Once the movement has been decided, we set some bit variables that are used to pilot the motors and represent direction and speed of the robot.

We use 3 more bits:

- ahead/turn:
  - 0 go straight on
  - 1 turn
- direction:
  - 0 right
  - 1 left
- speed:
  - 0 slow
  - 1 fast

S2	S1	S0	M2	M1	M0
0	0	0	change direction if steering		
0	0	1	1	1	0
0	1	0	keep steering or steer right if going straight		
0	1	1	1	1	1
1	0	0	1	0	0
1	0	1	0	-	-
1	1	0	1	0	1
1	1	1	find the lost line, steering if not at startup		

Note that there are some combinations (000, 010, 111) representing unusual cases (that is, forks, crossroads, 'S' bends) treated in a particular way.

LFR Software Architecture

PWM Update

After M2, M1 and M0 are set, we can use them to change the movement of the LFR (task 2).

In particular we have to assign the right values to the PWM0L and PWM0H registers which control the duty cycles of PWM signals of left motor and right motor respectively:

- if the robot has to go ahead we put the same value (PWM_MEAN) in both of them;
- if the robot has to steer slowly we decrease the speed of the bend-side motor by a quantity equal to PWM_STEP (we don't change the other motor);
- if the robot has to steer quickly we act as previously but decreasing by 2 * PWM_STEP.