Acrob (Arduino)

Minirobot ,,Franta"

(Igor Lacík & František Nagy)

Overview:

Franta in action is found here: http://www.youtube.com/watch?v=rqoO1gnbeUE

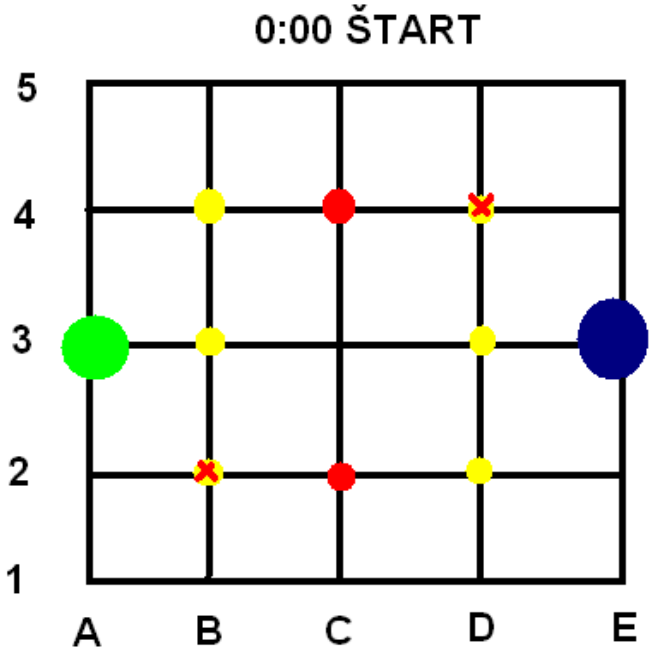
In the summer semester at 2012 I (Igor Lacík) and my friend (František Nagy) were visiting the courses of Algorithms of AI robotics led by Mgr. Pavel Petrovič, PhD. When we were presented with the opportunity to program a minirobot for the robotic competition Istrobot, it just felt stupid not to try out our skills.

We chose the http://en.wikipedia.org/wiki/Arduino, because I had like half an hour more experience in programming in it, than programming the http://virtuallab.kar.elf.stuba.sk/robowiki/index.php?title=SBot_2.0 SBot, what felt like a reasonable argument ☺.

We chose to compete in the category "V sklade kečupu". There are more information about Istrobot 2012 http://www.robotika.sk/contest/2012/index.php

I shall explain the rules very quickly:

1. You start either on the green, or the blue spot.
2. Four ketchup cans are placed on the board. Two of them have fixed positions on the red marks, the other two are placed opposite each other randomly on the yellow spots. If there is a can on B3, then there's gonna be one on D3, if there is a can on B2 it looks like it does on the picture.
3. Each of the robots starts at the same time and have to bring as many cans as possible to their line. If your robot is the one starting on the green spot, you have to bring the cans to the A line.
4. That is not it. If the other robot is able to steal your cans before the time is up, he gets the points.
5. The game lasts for 3 minutes, or until both players acknowledge, that nothing more is going to happen.
6. The winner is the robot with most cans.



In this document I am going to describe the process of programming the concrete robot Franta for the Arduino platform. By the way, it ended up on the first place with two other robots ☺

If you wanna watch the whole category compete, you want to visit http://www.robotika.sk/contest/2012/index.php and download the 9th video stream part 6. It works fine in VLC media player.

NOTE: I'm not gonna describe how to plug in the cables and how the sensors work. That can be found in the documentation on http://ap.urpi.fei.stuba.sk/sensorwiki/index.php/Acrob.

In this document you can find code for a line follower, tips for getting the compass work without knowing anything about it and other stuff.

In a lot of details I explain why we decided to perform some operation and try to think about what could have been done better.

To program an arduino robot you do not need to read the whole document, but you can search for useful tips that might save you some time. Enjoy.

Tactics:

1. Franta follows the line.
2. Franta knows where he is (uses something like Cartesian coordinate system).
3. When it's lost, it's able to find himself by using a compass.
4. Franta has implemented several steps inside it and decides which of them to use according to the state of the board (at least what he thinks the state is)
5. Stealing is easier then finding -> in the beginning, Franta tries to bring home as many cans as possible and then, for the rest of the game, it rides over the opponents line and searches for any cans. If it finds a can, it brings it home and continues stealing.

Hardware:

As I said, we got the body of the robot on Slovak Technical University and mostly just programmed it. But it seems appropriate to write down the hardware specifics, used sensors and ideas that might have improved the robot have we thought of them before coding.

Processor: ATmega328

Memory: 32kB

Frequency: 16Mhz

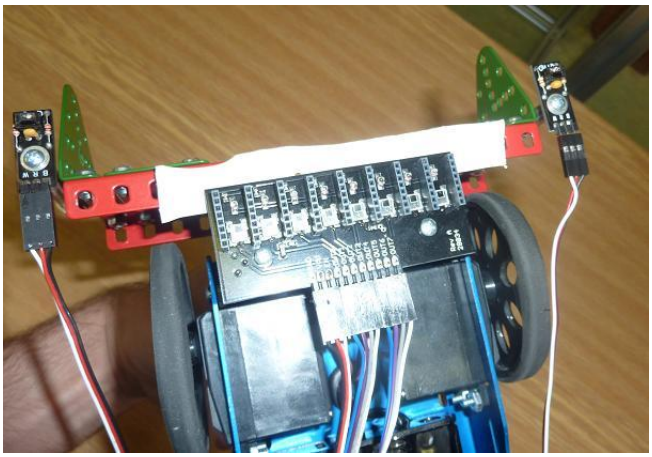
Driven by: 2 modified servos

Powered by: 4x AAA 1,5 V

Programmed in: Arduino C

The body of the robot is constructed with mercury, which worked very nice and looked a bit cooler then Lego robots ☺

Sensors:



Line sensor:

To follow the line we got a thing with 8 digital sensors to follow the line (as you can see on the picture).

The declaration at the beginning of the program code looks like this:

```
#define LINE_SENSOR(n) n+2;
```

The +2 means that the sensor uses the inputs from 3 to 10.

(!!!!between your sensors and another conductor there should be an insulator!!!)

A great suggestion when it comes to these line sensors:

We should have put them in the middle of the bottom part of the robot. When the robot performs an action turn left or right, and is carrying one or two cans, there is a chance that it's going to lose the line. When you check out the `void odbocVpravo()` code, you can see, that in order to turn right, the robot has to go a bit forward, to get on the crossroad, then it turns right and then it goes a bit more forward. All of this without checking the loss of the line.

If the sensors were in the middle of the bottom part of the robot, it would be much easier. The robot would just have to move to the crossing of the crossroad, turn 90 degrees in the way it wanted and it could continue the `void chodRovno()` function which takes care of not losing the line.

Additional line sensors:

You might have noticed the elegant turn that our robot does when parking a can (YouTube video, 10th second). We were trying to figure out how to make sure, that the robot will put his can directly on the line.

Ferro got the idea of providing the robot with additional line sensors. By putting them at the middle of the ending of our robots "antlers", it knows exactly when to do the elegant turn programmed in the procedure `void otocenie()`.

How to use the sensors?

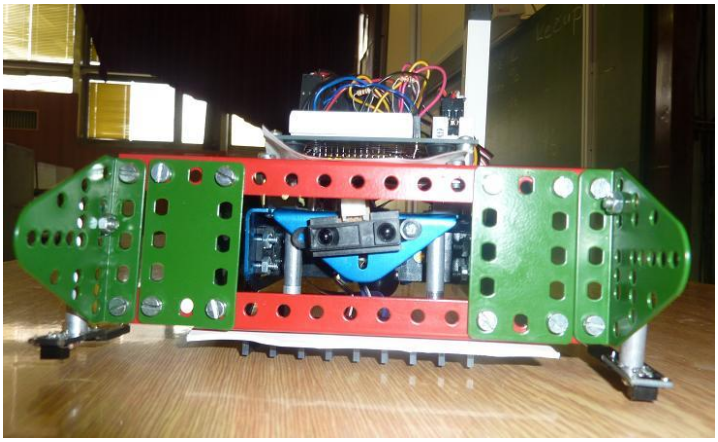
```
int black(long i) {

      return digitalRead(LINE_SENSOR(i));

}

// put the sensor as low as possible, otherwise the lightning can be a
//problem
```

Distance sensors:



We tried to save time wherever possible. By inserting the distance sensor (those cute 'eyes' on the picture) we were able to recognize, if Franta took a can on C2 or B2. If it did, it parked it on B1 and continued up instead of having to go to D2 and be sure that it took all the cans closest to it. (there will be more about tactics later)

Also, stealing wouldn't be effective, if the robot would not know if it took anything on the opponents line.

This is how we implemented the "check if something's close" method. It took a bit of testing, to find out what would work as often as possible.

```
int blizko() {

  int result = 1;

  for (int i = 0; i < 10; i++){

    result &= analogRead(distSensor) > 600;

    delay(1);

  }  return result;

}
```

Compass:

We knew, that our robot is losing its way and were quite afraid of leaving the Istrobot ashamed. So like one day before the competition we decided to add a compass.

That is the story of how Franta got a plastic chimney.

1. The first thing we found out was that the compass doesn't work very well when it's near to metal.
2. Second thing we found out was that the compass doesn't work very well even when it's on the chimney.
3. It worked better, so we started programming.

How to use the compass?

You do not have to know anything about the compass to use it.

At the bottom of our source code there is a comment FUNCTIONS and bellow that there are functions for the compass. All can be found in the documentation on Acrob wiki. You can use it to set "North" which in our case means front.

It didn't work really well, but we didn't have much time, so we found the average value of several measuring and believed it would work in the right time. It did ☺

```
void nastavSever() {
  compass();
  for (int j = 0; j < 5; j++)
    sever += compass();
  sever /= 5;}
```

Programming and source code:

Now that I explained why we chose needed sensors and how to use them, I'm gonna write some useful tips for you to start programming as fast as possible.

Setup:

In the setup part of arduino program you setup everything you want to use in the loop that should be set up :)

```
void setup() {
  LeftServo.attach(9);
  RightServo.attach(10);
}
```

Loop:

In the loop the actions are listed. This loop will cause the robot to turn clockwise until the batteries are gone or the robot is turned off.

```
void loop() {
  odbocVpravo();
  delay(200);
  chodRovno(); //use delay to state the time the robot should be going forward
  delay(200);
}
```

Tracking time:

If you wanna track time, you can use the function millis(). Declare a variable like this:

long startTime = millis();

Get the elapsed time by using this function:

```
long timeEllapsed() {
  return millis() - startTime;
}
```

And here is how you can stop the robot from turning infinitely after 60 seconds:

```
void loop() {
  odbocVpravo();
  delay(200);
  chodRovno();
  delay(200);
  if (timeEllapsed() > 60000) {
    while (1) {
      leftServo = 90;
      rightServo = 90;
    }
  }
}
```

Servos:

Franta uses modified servos. Normally, servos can be turned only 180 degrees and then there is a barrier that prevents them from moving more. When servos are used as wheels, the barrier is dropped.

This is how you set up servos:

```
#include <Servo.h>
Servo LeftServo;
Servo RightServo;
void setup() {
  LeftServo.attach(9);
  RightServo.attach(10);
}
```

The servos can gain values from 0 to 180. 0 to 90 which basically means that the wheel it is rotating clockwise in some speed, while the closer to 90 the value is, the slower the wheel turns. The moment it gets the value greater then 90, the servo is turning counterclockwise. I am sure you didn't understand any of that so it's like this:

```
leftServo == 90 // doesn't move
leftServo == 80 // moves slowly clockwise
leftServo == 10 //moves pretty fast clockwise
leftServo == 100 // moves slowly counterclockwise
leftServo == 180 //moves as fast as possible counterclockwise
```

Since the servos are placed opposite to each other, when you wanna go forward really fast, the values need to be:

```
(leftServo == 180) && (rightServo == 0)
```

To turn on a spot the values need to be:

```
(leftServo == 180) && (rightServo == 180)
```

It can get a bit confusing, so we wrote a simple procedure to control the servos:

```
void leftMotor(int n) {
  LeftServo.write(90+n);
}
void rightMotor(int n) {
  RightServo.write(90-n);
}
```

Now here is a little demonstration:

```
        rightMotor(FAST); leftMotor(FAST); //forward
        rightMotor(-FAST); leftMotor(-FAST); //backward
        rightMotor(-FAST); leftMotor(FAST); //turn around on a spot
```

Following the line:

How to follow the line?

It works like this:

The line sensor thing has 8 sensors. Looks like this: 1 2 3 4 5 6 7 8

The line is thick enough to be under two of the sensors.

Now the line is under the sensors 4 and 5. You know your robot is on the line. So you can go forward for a bit.

1 2 3 4 5 6 7 8

Now the line is under the sensors 5 and 6. The robot is still on the line, but it's best to be in the middle, so we have to move the robot to the left a bit.

1 2 3 4 5 6 7 8

Now the line is under the sensors 4 and 5. You know your robot is on the line. So you can go forward for while.

1 2 3 4 5 6 7 8

The source code of this procedure is here:

```
void chodRovno() {
  if (strateny()) { //if none of the 8 sensors detect a line, executes a process to find itself
    najdiSa();
  }
  int left = FAST, right = FAST;
  if (odchylkaVlavo()) {
    right = 0;
  }
  else if (odchylkaVpravo()) {
    left = 0;
  }
  leftMotor(left);
  rightMotor(right);
}


int odchylkaVpravo() {
  int result = 0;
  if (okrajVpravo()) result |= black(4);
  if (!okrajVpravo()) result |= black(5);
  result |= black(6);
  result |= black(7);
  return result;
}


int odchylkaVlavo() {
  int result = 0;
  if (okrajVlavo()) result |= black(3);
  result |= black(0);
  result |= black(1);
  if (!okrajVlavo()) result |= black(2);
  return result;
}
```

Keeping information about position:

It was very important to know where the robot stands in the map.

For this we declared variables where we save the point where the robot is heading and the direction in which he is.

The default values are X = 2, Y = 1 which basically means we are heading to B3 by the direction UP, in other words North.

```
int surX = 2;

int surY = 1;

char smer = 'U'; // U = up, L = left, R = Right, D = down
```

Travelling evolved from the `void chodRovno()` method. After declaring the variables for position we implemented the `void hladajKrizovatku()` method which basically performs `chodRovno()` until most of the line sensors aren't "black" == 1. The position is not changed yet, but only after Franta performs the `odbocVlavo`, `odbocVpravo`, or `prejdiKrizovatku` method.

Now that the basic functions for position tracking were implemented, the method for sending Franta on a concrete point had to be written. It is a simple recursive procedure that can be found in the source code. It is too long to be explained here, but I shall give a short example of how it works.

EXAMPLE 1:

Robot's on 2,1, smer = 'U'. We command it to go to 4,4.

Robot goes up until it's on the y == 4 position - 2,4.

Robot turns right and goes to 4,4.

Finding position after losing line:

Since the robot doesn't make any checks about other robots, it is probable, that it's gonna lose the line after crashing into another robot. We had to options.

1. Add a function that would recognize the opponent.
2. Add a function to get back on the game board.

A simple way to recognize the opponent would be to use the distance sensor. If the distance sensor would get an object in its sight, the robot would stop. If read by the distance sensor would start changing, Franta would know, that the opponent is moving and it's probably the opponent.

Even though it would be easy to implement, we thought it would make our robot a lot slower and so we decided not to add this functionality. After the competition, I think I would be implementing this for the next time.

So we decided to add the compass.

After finding out how it works (not amazingly well) we programmed a function that finds out if the robot is lost (`int strateny()`). Each calling of the method `chodRovno()` checks whether the robot is lost. If it is, the long process of finding itself is being executed.

These steps show how the process works:

1. Find out that you're lost (`void chodRovno() -> int strateny()`). Executes void najdiSa().
2. Find the line (`int najdiCiaru()`) which works like this: go forward for a second, go backwards for 2 seconds, go forward for a second, turn right and go forward for a second go backwards for a second until you find the line (`!strateny()`).
3. Find the next crossroad. The `void chodRovno()` method is taking care of not losing the line again (works line 9 out of 10 times).
4. When on crossroad, find North (causes the robot to make a one or two rotations)
5. When North is found, go up while possible (until not lost).
6. When on y = 4 (can't go up) turn left and go forward until possible.
7. The location should now be 0,4.
8. Known bugs: when on point 0,4, the robot tends to search for its position infinitely, but something like 2 out of 5 times it works ☺

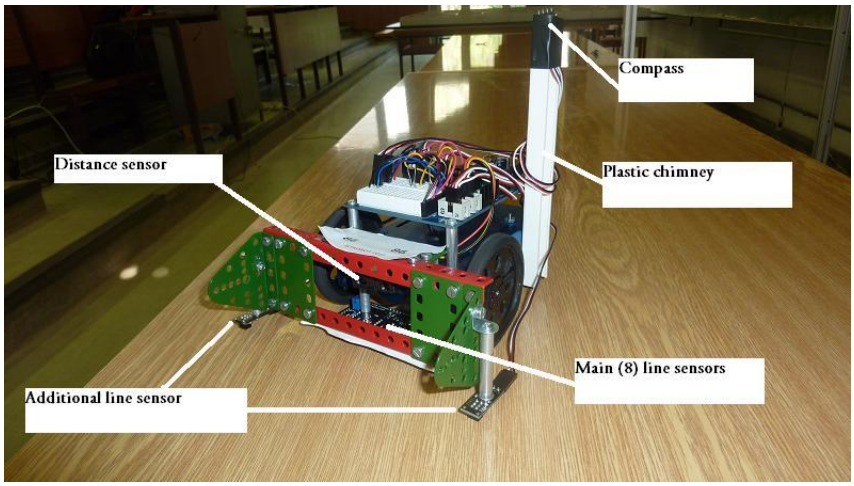
Tactics:

With all the control methods implemented we wrote down a list of steps that built the tactics for our game-play.

The method can be found as `void taktika1()`:

1. The robot starts heading up to y == 1, x == 2.
2. If it takes the can either on y == 1 x == 2 or y == 1, x == 1 then it parks one or two cans on y == 0, x == 1.
3. If it doesn't take the cans on mentioned locations, it continues its way to x == 3, y == 0.
4. After parking the cans, it checks the locations where the other two cans could be. If it finds anything, it parks them on its line.
5. For the rest of the game it searches over the opponents home line and if anything is found, it brings it home and continues the search.

Robots photo with description:



Last words:

Programming the robot was great fun. We never dreamed of winning. I think it would be a great waste if you were on FMFI and took the courses of Alghorithms for AI robotics and not programming a simple mini-robot like this.

I hope this document will help future students with some of their problems when programming a robot in Arduino. The source code used on the competition is linked below, I hope you will be able to reuse as much as possible even though some of it is a real mess ☺

The source code is here: http://ulozto.cz/xYU3aDu/robo-fero-igor-rar