2013

Universidad del Valle de México

LAUREATE INTERNATIONAL UNIVERSITIES®

1

2
Liliana Villamar
Alberto Vázquez 3
Juan Lebario

4

Universidad del Valle de México.

Team Description Paper

Festo Logistics League 2013

30/03/2013

[UVM INGENIERIA]

Just as automation needs have changed in recent years, along with the challenges we face, our team is renewing every day using new platforms and technologies that will allow us to face the "Festo Logistics League 2013" with the best results. The following paper describes our team history, along with the new platform and technologies we have implemented and are to be done.

1 Introduction:

Since we began our participation on National Robotic contests we had obtained achievements and great results, implementing every time better our programs and robotics features.

1.1 Team History

On 2011, one of our team members participated on the 7th Institutional Mini-Robotics Contest (MR7) and Robocup 5, obtaining the third place. Then, we participated on the 7th Automation challenge UVM-Festo, obtaining 1st and 2nd place.

On 2012, we participated on the 8th Institutional Mini-Robotics Contest (MR8) and Robocup 6, as well as on the 8th Automation challenge UVM-Festo obtaining the first place on both competitions.

Thanks to our efforts on national contests, we had the opportunity to participate on Robocup 2013 in Mexico City, wining the 3rd place.

1.2 ¿How we started?

1.2.1 The Software

1.2.1.1 Solving the Challenge

We started using Robotino-View as our first platform; we created a very simple algorithm to solve the challenge as shown below: (See Table 1).

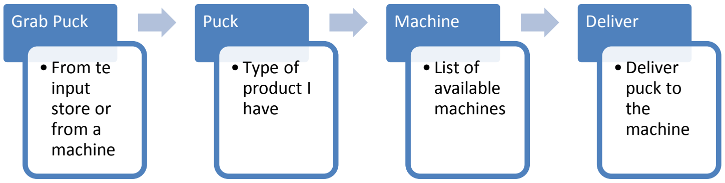

Grab Puck
• From te input store or from a machine
Puck
• Type of product I have
Machine
• List of available machines
Deliver
• Deliver puck to the machine

Table 1. Solving the challenge

To enhance the functionalities of Robotino View 2, we introduced Function-Blocks written on C++. This way we could solve algorithms easier and faster with Robotino View 2.

As an example, the code shown below was used to search stored products on our server:

```
if(Type_of_Product){
            switch(Product_I_Have){
                  case 0:
                        for(int i = 0;i<Number_of_Type_of_Product; i++){
                              if(Pocks[i]==Type_of_Puck){
                                    found = i+1;
                                    take =1;
                                    product =2;
                                    break;
                        }
```

```
                }break;
```

The same algorythm was used to search machines over the field.

The robot navigation was also controled by function block storing all posible directions over the field:

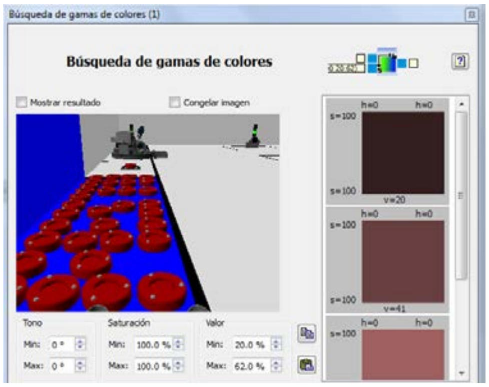
```
switch(Initial_Machine){
            case 0:
                   switch(Final_Machine){

                        case 0:
                        if(count==0){
                                    XOdom = VectorX[0];
                                    YOdom = VectorY[0];
                                    GOdom = VectorG[0];
                                    count++;

                              }else{
                              if(cont>=1){
                                       Initial_Machine = Final_Machine;
                                       count = 0;
                                       Position_Reached = true;
                              }}
                              break;
```

1.2.1.2 IMAGGE PROCESSING

As for Image processing we used the blocks available on Robotino-View:



Búsqueda de gamas de colores (1)

Búsqueda de gamas de colores

□ Mostrar resultado □ Congelar imagen

Tono Min: 0 ° Max: 0 °

Saturación Min: 100.0 % Max: 100.0 %

Valor Min: 20.0 % Max: 62.0 %

FIG. 1. IMAGE PROCESSING ON ROBOTINO VIEW 2

1.2.1.3 COMMUNICATION

The communication between Robotino's was also very easy, the information available to all robots was the status of all machines in the game, and the products stored at every machine. The tool used to communicate them, was the exchange data server included on Robotino View 2

1.2.2 HARDWARE

The robot implemented and /or used the following sensors or actors:

- Infra-Red
- Optical
- Camera

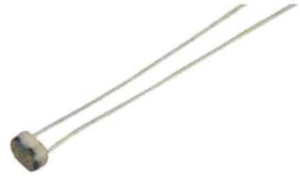
- Gyroscope
- Kicker
- Photo-electric sensors

1.2.2.1 *Photo-Electric Sensors*

One of the biggest challenges we faced was to recognize and process the signal from the lights on field, as they were very bright, so we implemented three sensors one for each LED of the light.

The implementation of this sensor, alowed us to receive correctly the signals from the lights.

Fig. 2. 10 ohms photoresistor



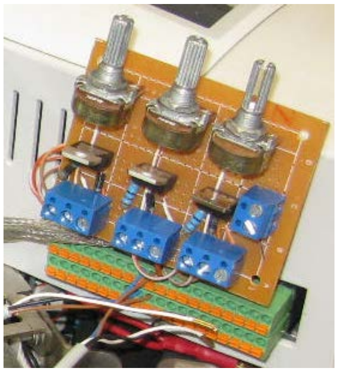


Fig. 3. photo-electric controller

2 Current work

From the experience we had on the last Robo-Cup, we are working on improvements on software and hardware, basically our work is centred on the following:

2.1 Software

We are no longer using the Robotino-View platform, as it has a lot of disadvantages in performance, and it has a very limited logic.

Our work will be focused on Java and OpenCV, using the robotino API to control every sensor and actor from the base.

2.1.1 Navigation

Our navigation is done through riels that exist on the field, so the robot can easily search a path to reach it's destiny.

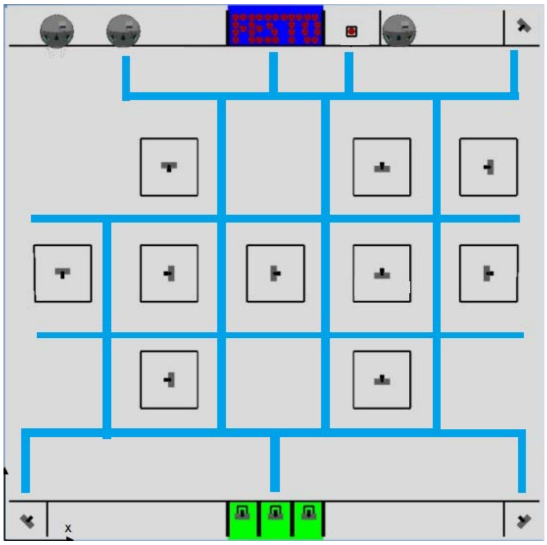


Fig.4. Virtual Riels on the Cord

The riels are also used to reduce the possibilities while calculating the route form A to B, and for this is used the A* algorithm.

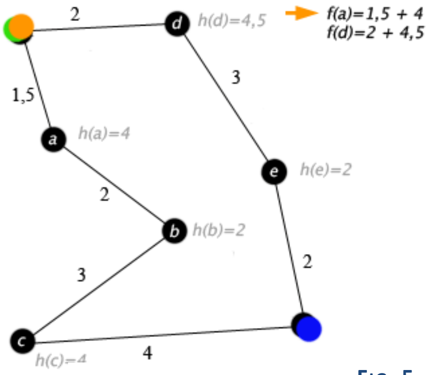


Fig. 5. A* Algorithm. Source: http://en.wikipedia.org/wiki/A*_search_algorithm

The

Navigation is controlled by the odometry and the gyroscope sensor on the robot, and we are looking to reduce navigation errors using OpenCv.

We control the omnidrive with the readings from the odometry, as we can give a function a point X,Y to reach, the robot will reach it over a holonomic navigation.

The holonomic navigation is given by the following Java code:

```
if(diferenciaX>tolerancia){
this.velPhi = (diferenciaPhi*this.velocidadEstandar)/diferenciaX;
this.velX = Math.cos(Math.toRadians(phiOdom))*velocidadEstandar;
this.velY = -Math.sin(Math.toRadians(phiOdom))*velocidadEstandar;
robot.setVelocity((float)velX, (float)velY, (float)velPhi);
}else{
robot.setVelocity(0, 0, 0);
continua=false;
}
```

2.1.2 Image procesing

For Image processing, we are using a Java wrapper for open-CV (JavaCv), we are currently in research for this tool, but we are planning to use it for navigation, puck segmentation and signal light processing.Comunication

Comunication between robots will be more fluid. Will be included positions in real time of every robot, as well as the status of every machine and products stored.

2.2 New Robot Improvements

2.2.1 The new kicker

We have designed a new kicker, so the robot never misses the puck, it is also more exact while delivering and taking it from the input store.

2.2.2 New photo-electric Carrier

The three photo-electric sensors are carried with more accuracy, and we can adjust the distance easier.

2.2.3 New Camera

We have implemented a new camera with better capacities.

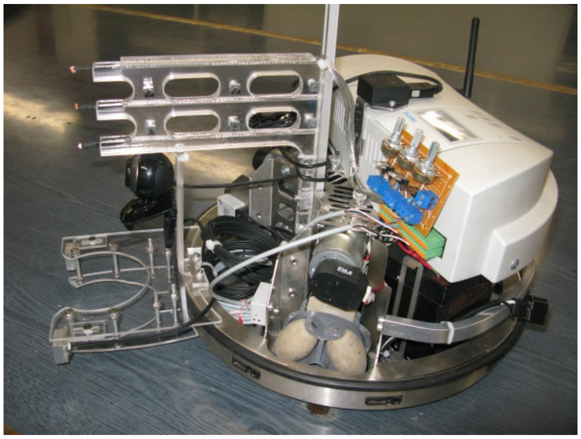


Fig. 6. Robotino with our new Puck Grabber, camera and light sensors

3 Conclussion

Our team is committed with new improvements every day, we consider we have made enormous advances since we began on Robotino Platform, and we are looking to implement new technologies that will allow us to have the best results in the Robocup 2013.