Remote Control of a Robotic Arm via Bluetooth

Gama H. Manuel, Villaseñor G. Isaí, Mondragón C. Carlos, Dr. Moisés Sánchez Adame, M.C. Adolfo Esquivel Martínez

Control remoto de un brazo robótico vía bluetooth

Recibido: enero 6, 2010
Aceptado: febrero 12, 2010

Palabras clave: Brazo robótico; control; bluetooth

Abstract:
This project involves the implementation of the IEEE 802.15.1 technology on the control of a two-axis robotic arm through two joysticks. This system implements a Bluetooth communication using two modules of Bluetooth, an Embedded Blue Transceiver AppMod EB500 Module and an Easy Bluetooth Parallax Module; as the main controllers, it uses two Parallax Propeller ProRPM microcontrollers. The user indicates with the joysticks how each of the three axes should move from a distance to another; these orders travel from an ADC circuit to the first microcontroller which are sent throughout a connection of Bluetooth communication to the second microcontroller, which decodes the commands and sends the information directly to the servomotors that move each axis. To avoid any undesired movement that was not commanded by the user using the joysticks, a digital PID controller was also implemented using two feedback potentiometers attached to two of the axes of the robotic arm. In conclusion, we have a Robotic Arm that is controlled by a Parallax Propeller microcontroller that uses Bluetooth technology to receive data from another Parallax Propeller Microcontroller that decodes the user's commands using two analog joysticks.

Keywords: Robotic arm; control; bluetooth

SINCE the dawn of humanity, a fundamental issue regarding the development of progress and evolution has been the ability to communicate with one another to progress.

The main principle of establishing communication is through speech and written language, which is based on the relationship between a sender, a message and a receiver. But technology today does not only refer to the communication of language, it tries instead to address a larger application: the data transmission. Given this need are emerging computer networks like the intranet, extranet and the Internet [1].

Therefore, the application that takes place within this project is the use of computer network communications technology as a demonstrative project in which the purpose is to be able to communicate two microcontrollers using Bluetooth technology, and interchange control instructions between them to manipulate a Robotic arm using analog joysticks controllers.

PRINCIPLES OF DESIGN

BLUETOOTH MODULES AND SERIAL COMMUNICATION

BLUETOOTH MODULES

Bluetooth wireless technology is a short-range communications technology. This was designed to replace the cables connecting portables and or fixed devices using radio frequency, while maintaining high safety standards. The key features of Bluetooth technology are robustness, low power and low cost. The Bluetooth specification provides a standard structure for a wide range of devices that can connect and communicate with each other wirelessly.
Bluetooth technology has achieved global acceptance such that any Bluetooth enabled device, almost everywhere in the world, can connect to other Bluetooth devices in range-wide [2].

Bluetooth wireless technology was designed to replace the cables connecting portables and or fixed devices using radio frequency

The modules that were used for the project were the AppMod EB500 Transceiver Embedded Blue Parallax Module, and the Easy Bluetooth Parallax Module, which both provide Bluetooth connectivity for microcontroller applications.
To simplify the Bluetooth connection between the two microcontrollers, we used only the Bluetooth Technology to transmit and receive the signals wirelessly, but the real protocol made within the technology was the traditional RS-232 Full Duplex Serial Communication.

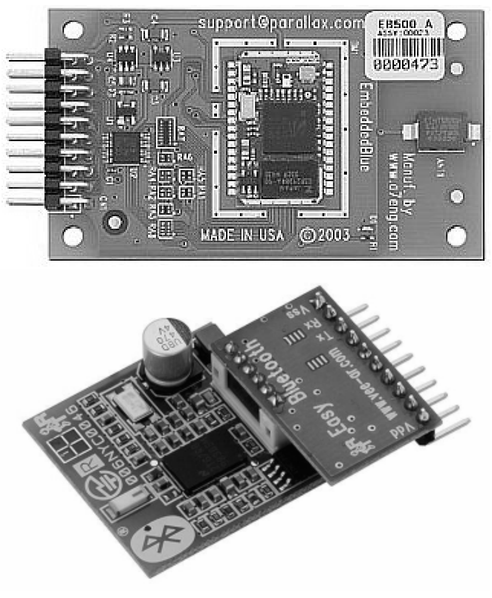


Figura 1. Embedded Blue Transceiver AppMod EB500 and Easy Bluetooth Parallax Module.

SERIAL COMMUNICATION

Serial communication is the most common low-level protocol for communicating between two or more devices. Normally, one device is a computer, while the other device can be a modem, a printer, another computer, or a scientific instrument such as an oscilloscope or a function generator. As the name suggests, the serial port sends and receives bytes of information in a serial fashion – one bit at a time. These bytes are transmitted using either a binary format or a text (ASCII) format. Over the years, several serial port interface standards for connecting computers to peripheral devices have been developed. These standards include RS-232, RS-422, and RS-485 – all of which are supported by the serial port object. Of these, the most widely used standard is RS-232, which stands for Recommended Standard number 232.

The current version of this standard is designated as TIA/EIA-232C, which is published by the Telecommunications Industry Association. However, the term “RS-232” is still in popular use, and is used in this article when referring to a serial communication port that follows the TIA/EIA-232 standard. RS-232 defines these serial port characteristics:

- The maximum bit transfer rate and cable length.

- The names, electrical characteristics, and functions of signals.
- The mechanical connections and pin assignments.

Primary communication is accomplished using three pins: the Transmit Data pin, the Receive Data pin, and the Ground pin. Other pins are available for data flow control, but are not required. [3]

The communication made by the two Bluetooth devices was segmented in a package that contains 6 words of data. The first, third, and fifth data were the data that indicated which axis was moving, while the second, fourth and sixth data were the data that indicated the movement of the servo motors. There were 3 main words that were reserved to contain the information of the movement of the servo. These main words were the numbers 0, 1 and 255; for example, if we wanted to move the servo that corresponded to the X axis, we send a 0 to understand that the reading from the ADCs corresponded to that specific axis, while we used a 1 for the Y axis and the 255 for Z axis for grip arm. The receiver had an algorithm to identify (decode) the command that was sent, and then, it transferred the data to a variable that was used to move that specific servo motor.

ANALOG JOYSTICKS CIRCUIT WITH ADCs

To control the robotic arm, a pair of two-axis joysticks was used, similar to analog joysticks controllers used for the controllers of video game consoles. Each joystick internally consists of an arrangement of two 100 K*Omega* potentiometers. The directional movements of the two axes of the robot were controlled by one joystick, while the second was used to control the arm grip mechanism that gave the arm the possibility to grab small objects of less than 400 grams of weight.

Because the output signals of the potentiometers of the joysticks are analog, we had to convert these signals in a digital form to be processed by the microcontrollers. In order to transform the analog signal to a digital signal, three ADC0831 integrated circuits were used.

DIGITAL PID CONTROLLER

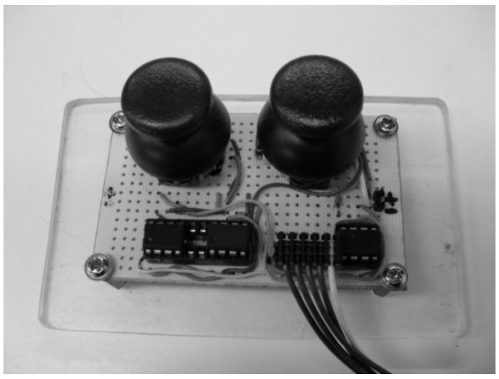


Figura 2. Control Joystick with ADC converters.

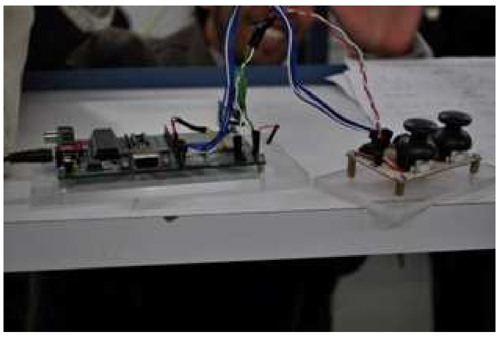


Figura 3. Final Joystick Controller.

PID THEORY

Proportional-Integral-Derivative (PID) control is the most common control algorithm used in the industry and has been universally accepted in industrial control. The popularity of PID controllers can be attributed partly to their robust performance in a wide range of operating conditions and partly to their functional simplicity, which allows engineers to operate them in a simple, straightforward manner. [4]

Proportional-Integral-Derivative control is the most common control algorithm used in the industry and has been universally accepted in industrial control

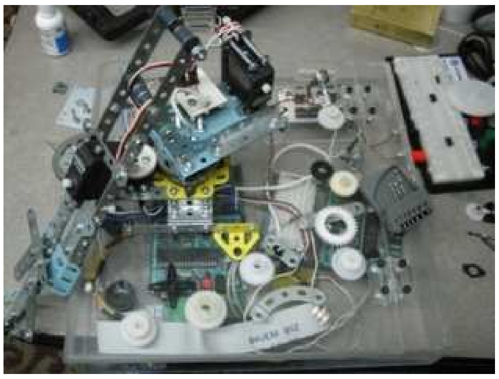


Figura 4. Implementing de two feedback potentiometers for use with the digital PID.

IMPLEMENTATION OF THE DIGITAL PID ON THE PROJECT

In order to have control of undesirable movements of the robotic arms that are not originated by the user's joysticks movement, a digital PID controller was implemented in the program of the microcontroller that controls the servomotors of the robotic arm. To have a feedback of the position of the main axis (X and Y), two feedback potentiometers were implemented in the gears of the two axes. The only limitation about this implementation was that the potentiometers don't turn 360 degrees, therefore, the two axis were limited to the physical limit of the two potentiometers.

To implement the PID control, as explained above, we used two potentiometers to read the set point of the last position of the arm. For example, the uC connected to the arm read the position data from the uC that had the Joysticks, and while the joysticks were moving, the PID method was not enabled. In the other case, when the values of any joystick were 127 (center position), a Boolean variable becomes true and then the uC that had the arm started to read the values on each of the two feedback potentiometers that were implemented for the feedback control. These values were stored in two variables and they become the set point of the PID. If any undesirable disturbance was made by any external force that had the sufficient force to move the position of the arm, the PID method will send the a code that will generate a movement to compensate that external movement or disturbance; if the variable becomes false again (this change could only occur if there is a change in the values of any joystick received by the uC) the PID becomes idle, and so on.

Figura 5. The Parallax Continuous Rotation Servo used in the project.

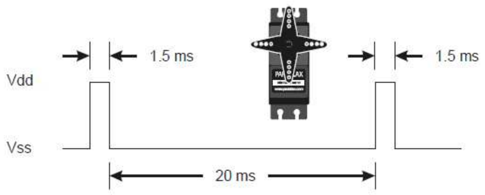


Figura 6. The servo motor is in the center position when the length of the pulse is exactly 1.5ms between the 20 ms pause.

PARALLAX CONTINOUS ROTATION SERVO

The Parallax Continuous Rotation Servo is a small servo motor that uses 4 to 6 V DC to operate, and a simple communication with a PWM capable device (such as the microcontroller) is done by connecting the PWM pin that each servomotor has (along with the other two pins: ground and Vcc).

The Parallax Continuous Rotation Servo is controlled through pulse width modulation, where the rotational speed and direction are determined by the duration of the pulse. In order for smooth rotation, the servo needs a 20 ms pause between pulses. Below is a sample timing diagram for a centered servo.

As the length of the pulse decreases from 1.5 ms, the servo will gradually rotate faster in the clockwise direction, as can be seen in the figure below:

Likewise, as the length of the pulse increases from 1.5 ms, the servo will gradually rotate faster in the counter-clockwise direction, as can be seen in the figure below [5]:

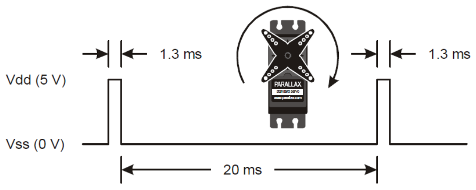


Figura 7. The servo motor is rotating to the right side when the length of the pulse is 1.3 ms or less between the 20 ms pause.

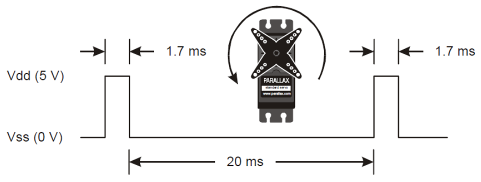


Figura 8. The servo motor is rotating to the left side when the length of the pulse is 1.7 ms or more between the 20 ms pause.

PARALLAX PROPELLER PRORPM MICROCONTROLLER

The microcontroller that was used in this project was the Propeller Rapid Prototyping Module (ProRPM), which is used as a high quality solution for project prototyping. The Propeller Microcontroller has eight cores inside, so it can process 8 simultaneous processes at the same time, while the programming languages that it supports are SPIN (A parallax programming language) and Assembly Language. For the purpose of the project, SPIN language was used because of the flexibility and ease of use it presents against assembly language. The ICs are all socketed, making it easy to replace parts if necessary. No extra hardware is required to program the microcontroller, only a DB-e Serial port cable is required to connect the microcontroller to a PC using the integrated RS- 232 interface.

MICROCONTROLLER PROGRAMS

PROGRAM OF JOYSTICK CONTROLLER

First, the program initializes with the constants to be used, as well as the output and input pins, the baud rate used to transmit the serial signals and the clock signal. Then, the variables to be used are specified in order to transmit and receive the data from each of the axes. Then, a call to use the object Full Duplex Serial is defined, which is an object library that will be used to make the serial communication. This library is property of Parallax. Then, the main program is defined, in which the Bluetooth communication is initialized, the variables needed are defined, and finally, the methods leer and sendValues are called in order to read the digital signals from the ADCs and to send the values thru the Bluetooth Module.

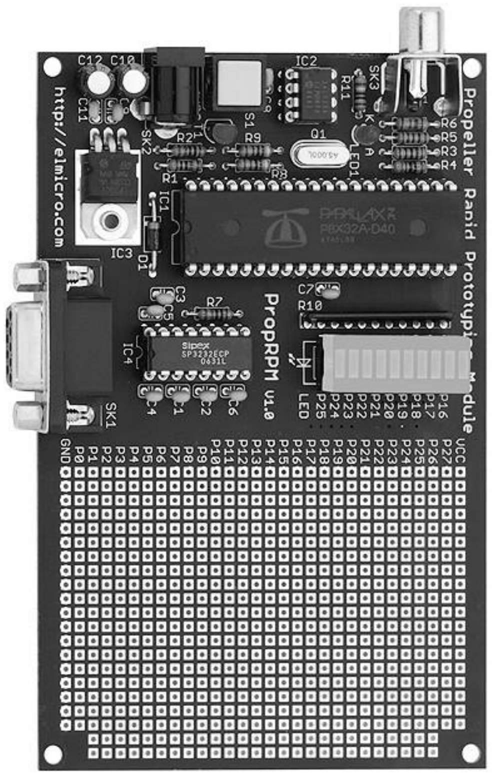


Figura 9. Propeller Microcontroller ProRPM.

```
PUB Main
'Inicializar puerto bluetooth
bluetooth.start(RX_PIN,TX_PIN,0,BAUD_RATE)
'Conectarnos de este micro
' al otro por bluetooth
'waitcnt(cnt + 160_000_000)
'bluetooth.str(string
("con 00:0c:84:00:75:c2"))
'bluetooth.tx(13)
'waitcnt(cnt + 400_000_000)
'Inicializar ADC0831's
dira[DX] := 0
dira[DY] := 0
dira[DZ] := 0
outa[CS] := 1
dira[CS] := 1
outa[CLK]:= 1
dira[CLK]:= 1
prevX := 0
prevY := 0
prevZ := 0
repeat
leer
'invertir eje Y
dataY := -1*dataY + 254
'Reservar los valores
0,1,255 if(dataX < 2)
dataX := 2
if(dataY < 2) dataY := 2
if(dataZ < 2) dataZ := 2
if(dataX > 254) dataX := 254
if(dataY > 254) dataY := 254
if(dataZ > 254)
 dataZ := 254 sendValues
```

Then, the method leer is defined, which will be the one to be used to read the signals from the ADCs. In order to have the highest efficiency possible, the connections Clock and Control had to be used in order to sample the data from the axes in the shortest time possible.

```
PRI leer
dataX := 0
dataY := 0
dataZ := 0
outa[CS] := 0
outa[CLK] := 1
outa[CLK] := 0
repeat 8
dataX <<= 1
dataY <<= 1
dataZ <<= 1
outa[CLK] := 1
outa[CLK] := 0
dataX += ina[DX]
dataY += ina[DY]
dataZ += ina[DZ]
outa[CS] := 1
```

Finally, a private method called sendValues is created, which does the primary function to send the data to the Bluetooth module towards the other microcontroller. The other microcontroller is the one that establishes the Bluetooth connection between the two; therefore, the data is automatically sent once the circuit is turned on.

```
PRI sendValues
IF prevX <> dataX
bluetooth.tx(0)
bluetooth.tx(dataX)
IF prevY <> dataY
bluetooth.tx(1)
bluetooth.tx(dataY)
IF prevZ <> dataZ
bluetooth.tx(255)
bluetooth.tx(dataZ)
prevX := dataX
prevY := dataY
prevZ := dataZ
```

PROGRAM OF ROBOTIC ARM MICROCONTROLLER

First, all the constants are defined:

-
- CLK is the clock to be used to time the ADC converters in order to synchronize them;
- CS is the control to be used in the ADC converters to determine the time that will be triggered in order to release the converted signal to the microcontroller;
- ADCx and ADCy, which are the inputs of the digital input of each of the ADC converters from the two feedback potentiometers used for the digital PID controller;
- centro (Center) is the constant that simplifies the center position of the feedback potentiometers used for the PID;
- PWM1, PWM2 and PWM3 are the output pins used to output the PWMs that control the servomotors;
- from gainX to dt, are constant needed for the operation of the digital PID controller;
- minimo is the constant that helps remember that the this value is the one needed for the servomotor to rotate complete to its left side continuously;

- centrar is the constant to remember the center PWM frequency to center the servomotor, or to stop it;
- escala is the resolution or scale used for the transformation of the analog signal that the potentiometers output to the ADC (this constant was calculated by dividing 255, which is the maximum data that the ADC could output because it is an 8 bit converter, divided by 8, which are the total bits used for one conversion);
- veinteMs is the constant used for the time off (is calculated by dividing the 80 MHz frequency in which the microcontroller runs by 20 microseconds, which is the time that is needed to control the time off of the servomotors);
- RX_PIN and TX_PIN are the Receive input and the Transmit output pins used for the Bluetooth communication;
- BAUD_RATE is the constant used for the Baud Rate that the Bluetooth communication uses;
- finally, the constants _clkmode and _xinfreq determine the clock used for the microcontroller to operate and the sample frequency used for the input pins.

Then, the variables to be used are defined:

- currentPointX and currentPointY are the current points used for the two digital PID controllers
- setPointX and setPointY are the set points used for the two digital PID controllers
- salidaPIDx, salidaPIDy and salidaPIDz are the PID outputs for each axis
- data is the variable used to determine the axis to move (it is used for the Bluetooth connection: 0 is used for the X axis, 1 for the Y axis and 255 for the grip arm
- dataX, data Y and data Z are the variables that represent the bytes that have the information of the movement of each of the joysticks used on the other microcontroller to position the servomotors
- salidaX, salidaY and salidaZ are the variables used for the value of the PWMs to output to each of the three servomotors
- from cogStack0 to cogStack4 are the array variables used for each new core that the microcontroller initializes to simultaneously make different processes.
- controlarX, controlar Y and controlar Z are the Boolean variables used to disable the PID digital controllers or to activate them

Then, the Full Duplex Serial library is called in order to use the Bluetooth connection. This same library is used in the other microcontroller to communicate the serial signal via Bluetooth.
Then the Main program is defined. In order to explain this part, which is very long, the explanations will be segmented in parts.
Main program code:

1. 1. The Bluetooth communication is initiated using the Full Duplex Serial library. Then, a waitcount is used to wait for the serial buffer to receive the Receiver Pin, the Transmitter Pin and the Baud Rate to use. After that, a string containing the MAC Address of the other microcontroller is used in order to make the first connection. Immediately after, a Carriage Return (represented by the number 13) is used to seal the connection, and finally, another waitcount is used to began the Bluetooth Communication.

```
PUB Main
bluetooth.start(RX_PIN,TX_PIN,0,BAUD_RATE)
'Conectarnos de este micro
' al otro por bluetooth
waitcnt(cnt + 160_000_000)
bluetooth.str(string
("con 00:17:a0:01:62:b0"))
'Conectar a MAC Address
bluetooth.tx(13)
'Mandar CR, Carriage Return
waitcnt(cnt + 160_000_000)
```

2. 2. The variables for the PWMs are initiated in the center position, in order for the robot to not move at all once the Bluetooth connection is first terminated.

```
'Inicializar Variables para salidas PWM
dataX := centro
dataY := centro
dataZ := centro
salidaX := 4000
salidaY := 4000
salidaZ := 4000
```

3. 3. The variables for the digital PIDs are replaced by the center position. Both PID outputs are turned to low position, and the variables controlar X, controlarY and controlarZ are set to false because the PID would not be activated at first.

```
currentPointX := centro
currentPointY := centro
setPointX := centro
setPointY := centro
salidaPIDx := 0
salidaPIDy := 0
controlarX := false
controlarY := false
controlarZ := false
```

4. 4. The 5 cores are initialized. One core for reading the two feedback potentiometers to use with the two digital PID controllers; one to move the X axis; on the move the Y Axis; one to move the Z axis, or the grip arm, and finally, one to calculate the two digital PID controllers.

```
'Inicializar 5 coprocesos mas,
' 1 para leer datos, 3
' PWM y 1 PID
cognew(leer(@currentPointX,@currentPointY),
@cogStack0)
cognew(mover(@salidaX,PWM1,@salidaPIDx,
@c ontrolarX),
@cogStack1)
cognew(mover(@salidaY,PWM2,@salidaPIDy,
@c ontrolarY),
@cogStack2)
cognew(mover(@salidaZ,PWM3,@salidaPIDz,
@c ontrolarZ),
@cogStack3)
cognew(PID(@currentPointX,@setPointX,@salida
PIDx,@currentPointY,@setPointY,@salidaPIDy),
@cogStack4)
```

5. 5. The main loop starts. In this loop, the Bluetooth connection is used to determine the bytes used for the axis detection and for the value that the joystick is making. The first byte, Data, is used to determine which axis is next to move. Then, the second byte, which could be dataX, dataY or dataZ, is used to determine the position of the joystick. If the position that the joystick sends to this microcontroller is a center position ± 1, then the digital PID controller is activated for the axis which the data came, if not, then the PID is disabled and the output PWM for each axis is sent to the servomotors. This logic repeats for each of the three axes.

```
'Loop principal repeat
data := blueTooth.rx
if (data == 0)
'Si se recibe un 0 entonces
' el siguiente byte es el valor
' del eje de las X
dataX := blueTooth.rx
if (dataX == centro OR
dataX == centro+1 OR
dataX == centro-1)
controlarX := true
setPointX := currentPointX
salidaX := dataX*escala
else
controlarX := false
salidaX := dataX*escala
if (data == 1)
'Si se recibe un 1 entonces el
' siguiente byte es el valor
' del eje de las Y
dataY := blueTooth.rx
if (dataY == centro OR
dataY == centro+1 OR
dataY == centro-1)
controlarY := true
setPointY := currentPointY
salidaY := dataY*escala
else
controlarY := false
salidaY := dataY*escala
if (data == 255)
'Si se recibe un 255 entonces el
' siguiente byte es el valor
' del eje de las Z
dataZ := blueTooth.rx
salidaZ := dataZ*escala
```

Then we create the method leer , which helps determine the position of the two feedback potentiometers and activates the two ADC converters used for the analog-to-digital conversion. This method is very similar to the method named the same on the other microcontroller that has the two joysticks implemented.

```
PUB
leer(valorX,valorY)| tempX,tempY
'Inicializar ADC0831's
dira[ADCx] := 0
dira[ADCy] := 0
outa[CS] := 1
dira[CS] := 1
outa[CLK]:= 1
dira[CLK]:= 1
repeat
tempX := 0
tempY := 0
outa[CS] := 0
outa[CLK] := 1
outa[CLK] := 0
repeat 8
tempX <<= 1
tempY <<= 1
outa[CLK] := 1
outa[CLK] := 0
tempX += ina[ADCx]
tempY += ina[ADCy]
outa[CS] := 1
long[valorX] := tempX
long[valorY] := -1*tempY + 255
'waitcnt(cnt + 80_000_000)
```

Then the method mover is defined. This method is used to calculate the PWMs used for each servomotor.

```
PUB mover(addressSalida,PWM,addressPID,
controlar)| tiempo
outa[PWM] := 0
dira[PWM] := 1 repeat
if long[controlar] == true
tiempo := long[addressSalida] +
long[addressPID] 'if long[controlar] == false
tiempo := long[addressSalida]
outa[PWM] := 1
waitcnt(cnt + minimo + tiempo)
outa[PWM] := 0 waitcnt(cnt + veinteMs)
```

Finally, we have the digital PID controller method which calculates the two digital PID controllers used to stabilize the system when undesirable movements occur to the robotic arm that is independent from the user's commands of the joysticks.

```
PUB
PID(cur_posX,set_posX,outputX,
cur_posY,set_pos Y,outputY)|
pre_errorX,cur_errorX,pre_errorY,
cur_errorY,        e, P, I, D
pre_errorX := 0
cur_errorX := 0
pre_errorY := 0
cur_errorY := 0
repeat
cur_errorX := long[set_posX] -
long[cur_posX]
P := gainX * cur_errorX
I := gainXi * cur_errorX * dt
e := cur_errorX - pre_errorX
D := gainXd * e / dt
pre_errorX := cur_errorX
long[outputX] := 0 - P - I - D
cur_errorY := long[set_posY] -
long[cur_posY]
P := gainY * cur_errorY
I := gainYi * cur_errorY * dt
e := cur_errorY - pre_errorY
D := gainYd * e / dt
pre_errorY := cur_errorY
long[outputY] := P + I + D
waitcnt(clkfreq / 1000 * dt + cnt)
```

RESULTS

First of all, once the two microcontrollers are turned on, the Propeller microcontroller that controls the movement of the robot connects itself with the other microcontroller via Bluetooth. In order to do that, the MAC address of the microcontroller that uses the analog joysticks has already been written on the communication protocol on the first microcontroller, and this allows the two microcontrollers to start communicating with each other. The operating mode of the robotic arm is then specified by the commands that the user wants using the joysticks control shown in Figure 1. The analog signals from these potentiometers are converted to a digital signal by the ADC0831s. These digital signals are interpreted by the Propeller microcontroller ProRPM, which processes the signals and codifies them to a serial signal that contains the movement of the two-axes of the robot arm, and the robot arm grip. This serial information is then used to transmit the control signals to the other Propeller Microcontroller via the Easy Bluetooth module, which is used as a replacement of a serial RS-232 cable. After the serial signal is directed to the other microcontroller that controls the servomotors of the robot via Bluetooth, this second microcontroller decodes the serial signal using the AppMod EB500

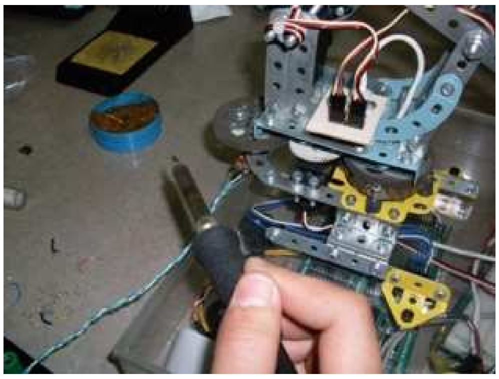
Transceiver Embedded Blue Parallax Module.

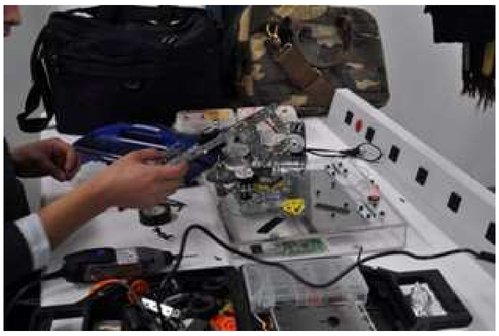
In order to know which axis has to move, in the serial code a byte is used to distinguish them. In the codification developed, number 0 was used to distinguish the X-axis, number 1 for the Y-axis, and number 255 for the arm grip or Z-axis. With the interpretation of this data, the servomotors of the robotic arm will move following the commands given by the user from any point, provided within the scope range of Bluetooth modules (which in the most positive case will be a 30 meters ratio).

If the robotic arm is moved undesirably by a force different from the signal of the joysticks (this could include moving the arm manually with a hand or putting more than 400 grams of weight on the grip arm), a digital PID controller is activated on the microcontroller that controls the servomotors and tries to stabilize the system to the last position that the user send using the joysticks. Two feedback potentiometers were used in joint with the Digital PID Controller, and they were implemented in the gear mechanism of the two main axes, X and Y. It is very important to point- out that the PID controller will always be enabled once the robot is not receiving signals from the joysticks because of the relative difficult implementation that a PID controller would have been if we wanted to avoid undesirable movements during the user's movement of the joysticks and therefore, the undesirable movement of the robotic arm during its normal human controlled movement.

CONCLUSION

Diverse knowledge was learned from this project, which was an inter-course project that joined the following courses: Computer Networks, Interface Design and Signals and Systems. The contents of the courses were used almost totally, but the implementation of the microcontrollers programs was a knowledge that was learned separately from these contents. The project did achieved the original objectives, and it also surpassed them because in the original design, a computer was used as an interface instead of one of the Bluetooth modules and the PID digital controller was not concerned also, but the implementation of another Bluetooth module and the PID digital controller were encouraged by the University's professors, so they were then implemented, but it also added more difficulty to the overall performance of the robotic arm.





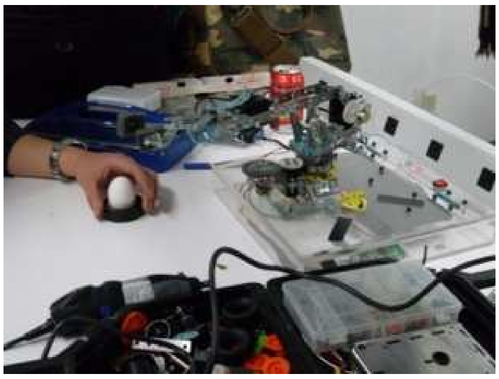
In conclusion, the robotic arm has a robust and fast response to the user's commands using the Bluetooth modules and the microcontrollers. Also, the PID digital controller response is very accurate, so the stability of the system is achieved in a high percentage. For the mechanical part, the use of pre- machinated construction metal sets helped to assembly the robotic arm with little adjustments, while the use of the continuous servo motors rated at 5V DC helped for the simple implementation of the movement of the arm; if DC motors and un- machinated parts were used, the difficulty to achieved the project would have been more difficult and time-consuming (maybe one month more of time would have been needed to use both).

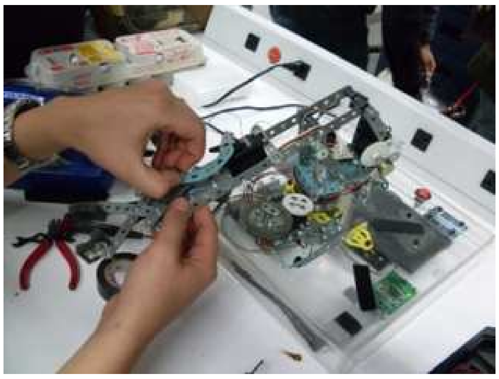
APPENDIX

Figures about the project development

ACKNOWLEDGMENT

Special thanks to Dr. Moises Sanchez Adame and the entire group of teachers at CETYS Universidad Tijuana, Prof. Arturo Escoto and Prof. Ricardo Martinez for their helpful





support during this project. Finally we thank our colleagues for the great help they gave us.

Bibliografía

[1] Johnatan Zavaleta Milla. (August 20th, 2006) Wireless Technology [On line]

[2] B. Hopkins, R. Antony, Bluetooth for Java USA 2003.

[3] http://www.mathworks.com/access/helpdesk/help/toolbox/instrument/f20-63744.html

[4] http://zone.ni.com/devzone/cda/tut/p/id/3782

[5] http://www.parallax.com/Portals/0/Downloads/docs/prod/motors/900-00008-CRServo-v2.0.pdf

Acerca del autor o autores

Gama H. Manuel, Villaseñor G. Isaí, Mondragón C. Carlos, Dr. Moisés Sánchez Adame, M.C. Adolfo Esquivel Martínez
CETYS Universidad, CITEDI-IPN Tijuana, Baja California, CP-22550. TEL: +(664)9031833, correo-e: ingenieromanuelgama@hotmail.com, realktm@gmail.com, moises.sanchez@cetys.mx, adolfo.esquivel@cetys.mx