7 Anexos

7.1 Anexo A: Sistemas Electrónicos

Unifilar del circuito eléctrico y protecciones eléctricas

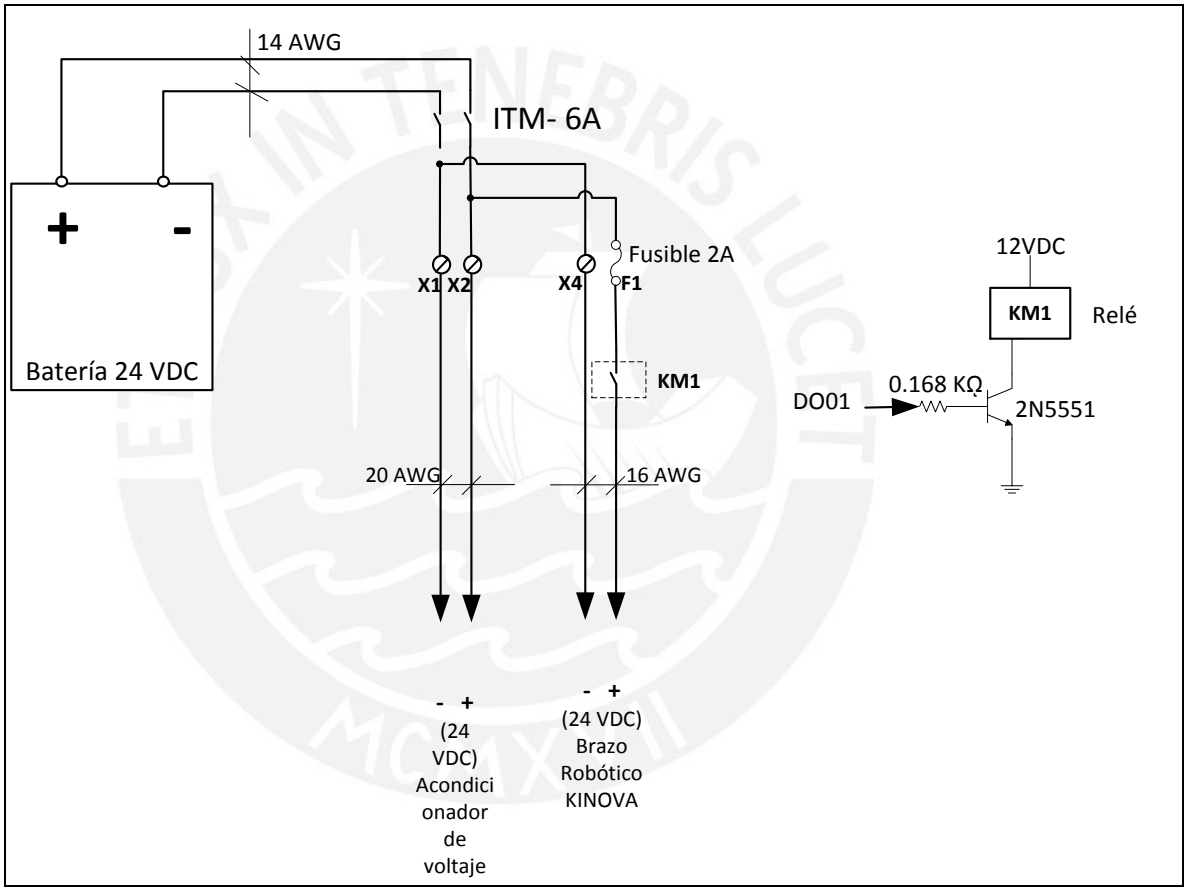
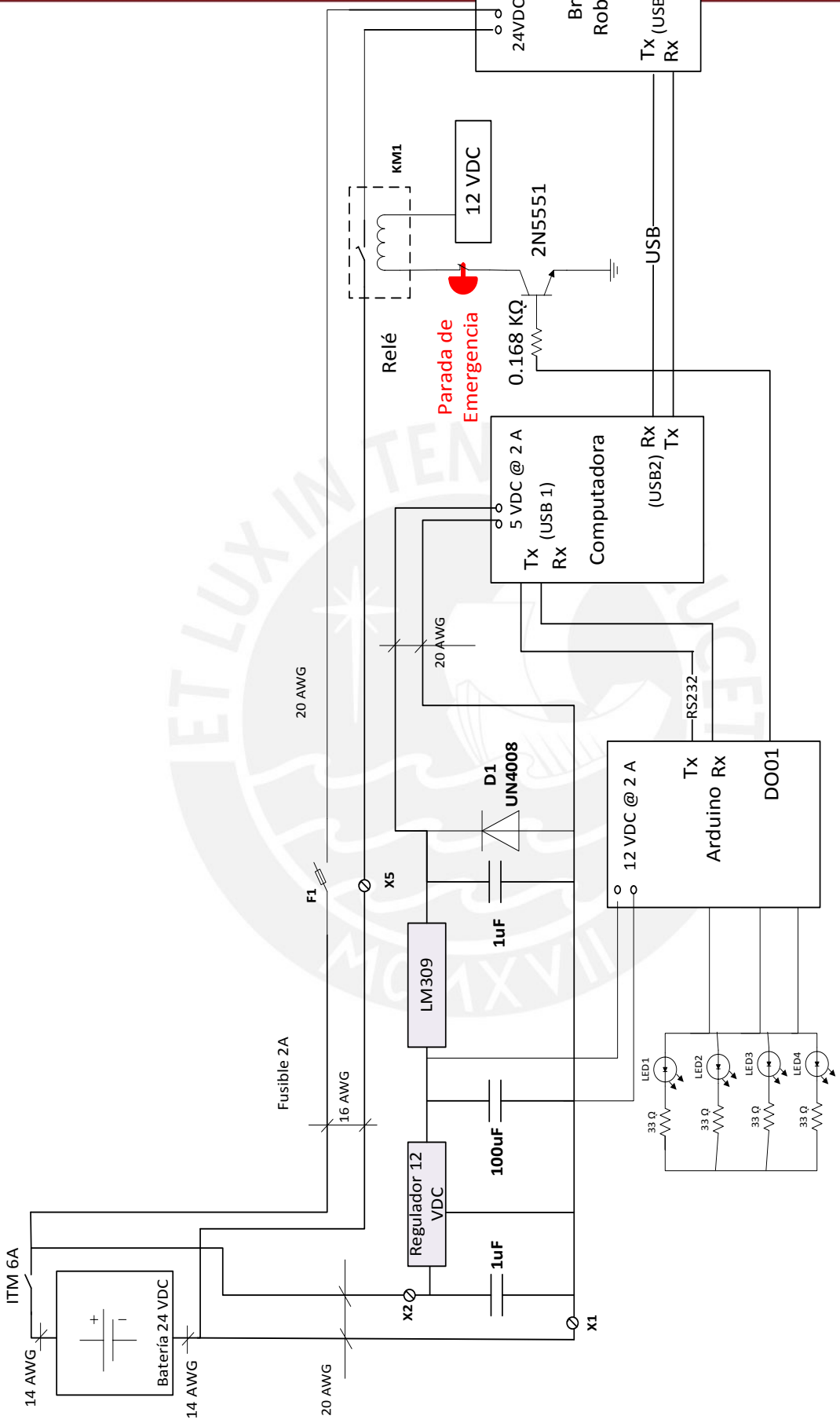
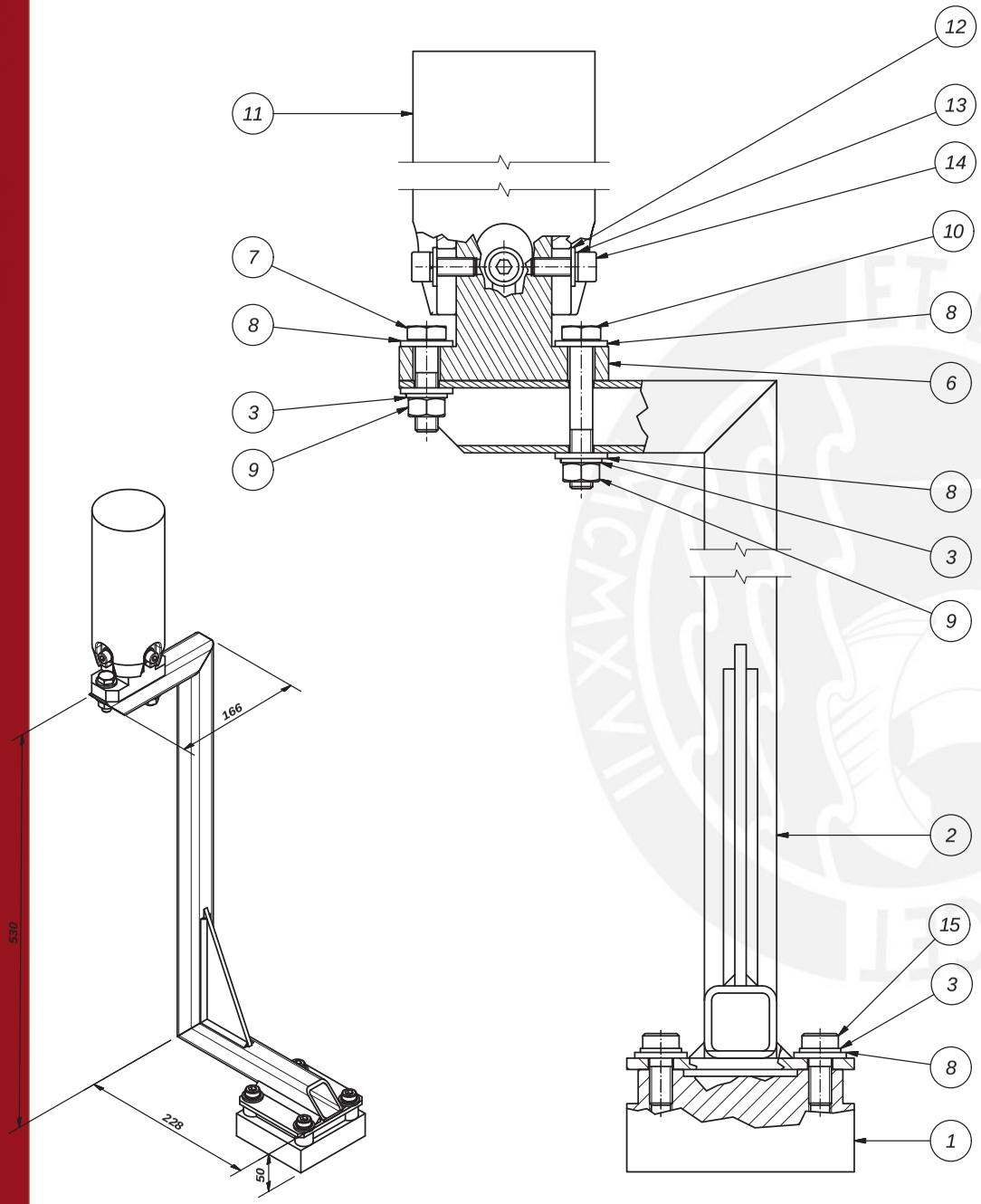


DIAGRAMA DE CONEXIONES ELECTRICAS DEL SISTEMA



7.2 Anexo B: Planos de Piezas y Ensambles

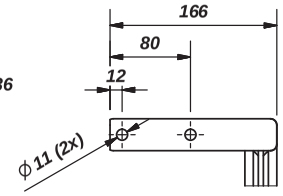


POS.	CANT.	DESCRIPCION	NORMA	MATERIAL	OBSERVACIONES
15	4	Perno de cabeza hexagonal - M10 x 25	DIN 6912		Grado 5
14	4	Perno de cabeza circular - M8 x 20	ISO 4762		Grado 5
13	4	Arandela de presión M6	DIN 128		Grado 5
12	4	Arandela Plana M6	DIN 6796		Grado 5
11	1	Robot		CF	Referencial
10	1	Perno de cabeza hexagonal - M10 x 65	DIN 931-1		Grado 5
9	2	Tuerca hexagonal - M10	ISO 4032		Grado 5
8	8	Arandela plana M10	DIN 6796		Grado 5
7	1	Perno de cabeza circular - M10 x 40	DIN 931-1		Grado 5
6	1	Mástil de brazo		AA 6063	Aluminio
3	6	Arandela de presión M10	DIN 128		Grado 5
2	1	Soporte de brazo robótico		ASTM A36	Parte soldada
1	1	Reductor con agujeros		Fundición	(Silla de ruedas)

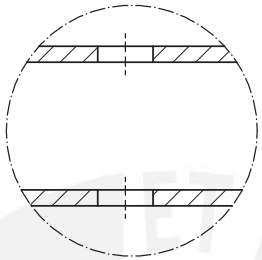
PONTIFICIA UNIVERSIDAD CATOLICA DEL PERU
SECCION INGENIERIA MECANICA

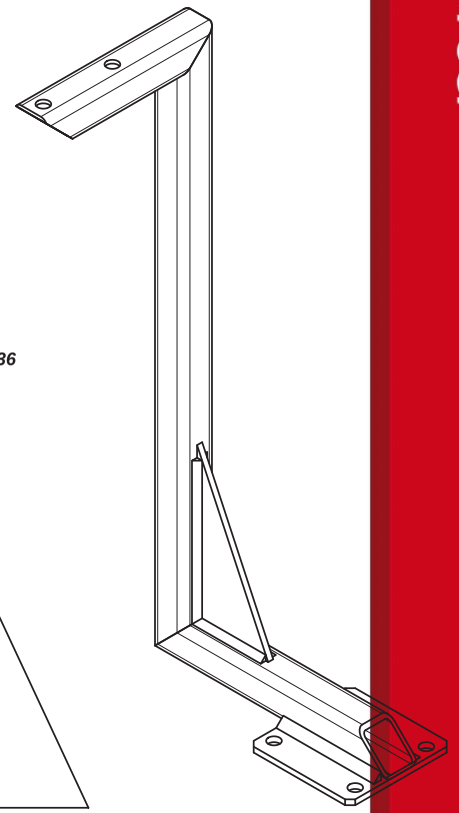
METODO DE PROYECCION	Maestría en ingeniería Mecatrónica - Becas CONCYTEC Soporte de brazo de robot	ESCALA 1 : 5
DIBUJADO POR:	César Coasaca	FECHA: 21.10.2015
APROBADO POR:		LAMINA: A3 - 1

ESTRUCTURA
TUBO CUADRADO 1-1/4 x 1-1/4 x 1/8 ASTM A36

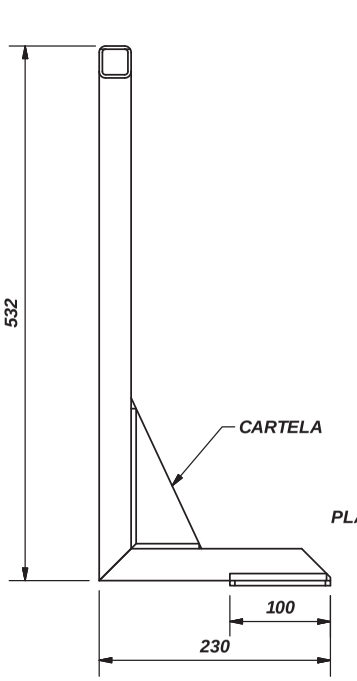


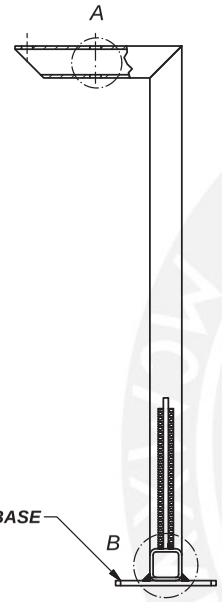
DETALLE A
(1:1)



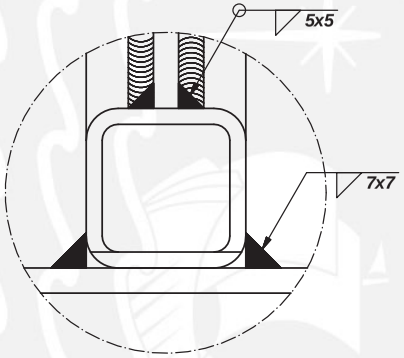


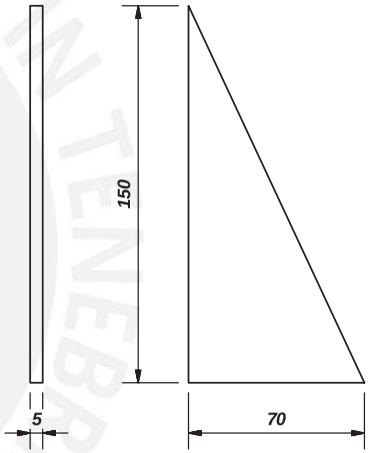
CARTELA
PLANCHA LAC t=5mm ASTM A-36



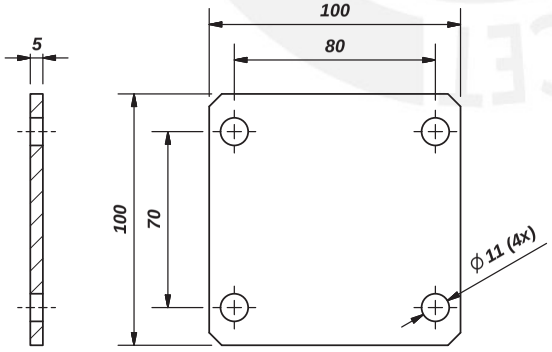


DETALLE B
(1:1)



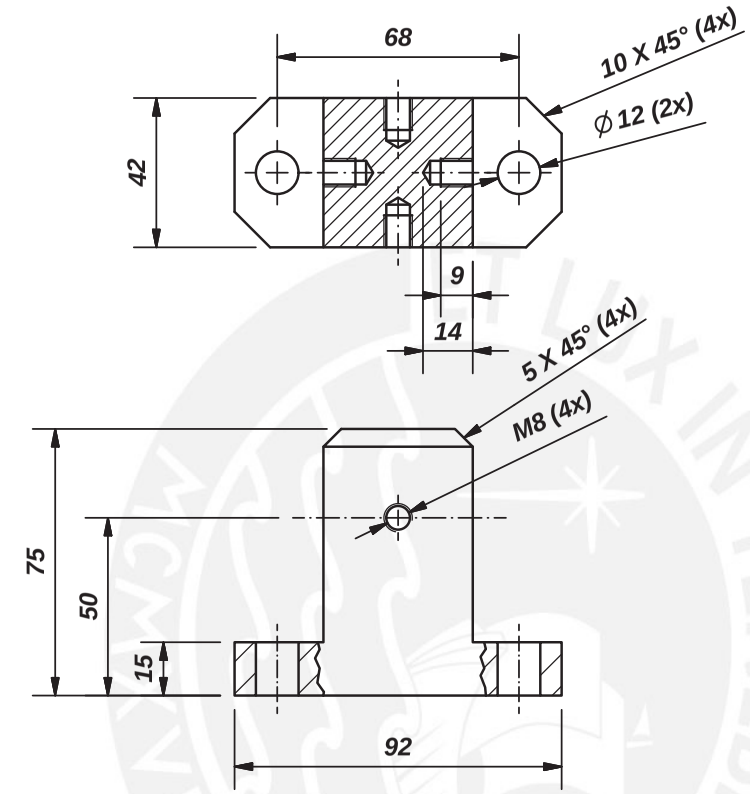


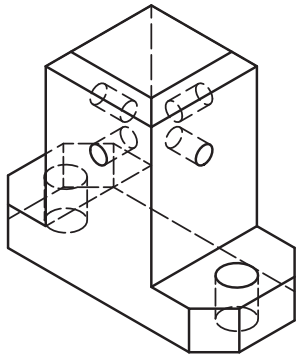
PLACA BASE
PLANCHA LAC t=5mm ASTM A-36



TOLERANCIAS DIMENSIONALES SEGUN DIN 7168							
GRADO DE EXACTITUD	Más de 0,5 hasta 3	Más de 3 hasta 6	Más de 6 hasta 30	Más de 30 hasta 120	Más de 120 hasta 400	Más de 400 hasta 1000	Más de 1000 hasta 2000
MEDIO	±0,1	±0,1	±0,2	±0,3	±0,5	±0,8	±1.2

ACABADO SUPERFICIAL	TOLERANCIA GENERAL SEGÚN DIN 7168 MEDIO	MATERIAL INDICADO
PONTIFICIA UNIVERSIDAD CATOLICA DEL PERU SECCION INGENIERIA MECANICA		
METODO DE PROYECCION	Maestría en ingeniería Mecatrónica - Becas CONCYTEC SOPORTE DE BRAZO ROBOTICO	ESCALA 1 : 5
DIBUJADO POR:	César Coasaca	FECHA: 22/10/2015
APROBADO POR:		LAMINA A3





ACABADO SUPERFICIAL	TOLERANCIA GENERAL	MATERIAL
▽	SEGÚN DIN 7168 MEDIO	AA 6066

PONTIFICIA UNIVERSIDAD CATOLICA DEL PERU
SECCION INGENIERIA MECANICA

METODO DE PROYECCION	Maestría en ingeniería Mecatrónica - Becas CONCYTEC	ESCALA
	MÁSTIL DE BRAZO	1 : 2
DIBUJADO POR:	César Coasaca	FECHA: 22.10.2015
APROBADO POR:		LAMINA: A4 - 3

TOLERANCIAS DIMENSIONALES
SEGUN DIN 7168

GRADO DE EXACTITUD	Más de 0,5 hasta 3	Más de 3 hasta 6	Más de 6 hasta 30	Más de 30 hasta 120	Más de 120 hasta 400	Más de 400 hasta 1000	Más de 1000 hasta 2000
MEDIO	±0,1	±0,1	±0,2	±0,3	±0,5	±0,8	±1.2

7.3 Anexo C: Programas

Programa General

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Emotiv;
using System.Windows.Forms;
using System.Threading;
using System.Net;
using System.Net.Sockets;
using Encog.Neural;
namespace DTF_Emotiv;

{

    public class Program
    {
       const int muestra=128;
       const int entrena = 1;
       const int umbral = 30;



        EmoEngine engine;
        int userID = -1;
         static int Comando =0;

        static double[,] O1_RAW = new double[entrena, muestra];
        static double[,] O2_RAW = new double[entrena, muestra];
        static double[,] P7_RAW = new double[entrena, muestra];
        static double[,] P8_RAW = new double[entrena, muestra];
        static double[,] SUMA_RAW = new double[entrena, muestra];
        static double[,] PROMEDIO_RAW = new double[entrena, muestra];

        static double[,] O1_Neutral = new double[entrena, muestra];
        static double[,] O2_Neutral = new double[entrena, muestra];
        static double[,] P7_Neutral = new double[entrena, muestra];
        static double[,] P8_Neutral = new double[entrena, muestra];
        static double[,] SUMA_Neutral = new double[entrena, muestra];

        static double[,] Entrena_Comando1 = new double[entrena, muestra];
        static double[,] Entrena_Comando4 = new double[entrena, muestra];
        static double Absoluto1 ;
        static double Absoluto4;

        static double[,] PROMEDIO_Neutral = new double[entrena, muestra];

        static double[] absoluto_O1 = new double[muestra];
        static double[] absoluto_O2 = new double[muestra];
        static double[] absoluto_P7 = new double[muestra];
        static double[] absoluto_P8 = new double[muestra];
        static double[] SUMA_absoluto = new double[muestra];
        static double[] PROMEDIO_absoluto = new double[muestra];
```

```
         static void Main(string[] args)
        {


        /*    const string MyValidPassword = "C6H12O6h2so4";
            Kinova.API.Jaco.CJacoArm Jaco_m = new
Kinova.API.Jaco.CJacoArm(Kinova.DLL.SafeGate.Crypto.GetInstance().Encrypt(MyVa
lidPassword));
            Kinova.DLL.Data.Jaco.Config.CClientConfigurations config = new
Kinova.DLL.Data.Jaco.Config.CClientConfigurations();
          */

             // Thread comunicacion = new Thread(Comunicar);// se define el
hilo de comunicacion con programa de BMI

                string sUserChoice = "";
                Program p = new Program();


            while (true)
            {
                if (Console.KeyAvailable)
                {
                    sUserChoice = Console.ReadLine();
                    if (sUserChoice.ToLower() == "x")
                    {
                        break;
                    }
                }


////////////////////////////////////////////////////////////////////////////
                    ////////////////////////  ENTRENAMIENTO
NEUTRAL//////////////////////////////////
                Console.WriteLine("Presione Enter para iniciar lectura de modo
neutral");
                Console.ReadKey();


                for (int n = 0; n < entrena; n++)
                {
                    p.Run(O1_RAW,O2_RAW,P7_RAW,P8_RAW ,n);
                    System.Threading.Thread.Sleep(1000);
                    p.Run(O1_RAW, O2_RAW, P7_RAW, P8_RAW, n);

                    for (int i = 0; i < muestra; i++)
                    {
                        O1_Neutral[n, i] = O1_RAW[n, i];
                        O2_Neutral[n, i] = O2_RAW[n, i];
                        P7_Neutral[n, i] = P7_RAW[n, i];
                        P8_Neutral[n, i] = P8_RAW[n, i];
                        SUMA_Neutral[n, i] = O1_RAW[n, i] + O2_RAW[n, i] +
P7_RAW[n, i] + P8_RAW[n, i];
                        PROMEDIO_Neutral[n, i] = SUMA_Neutral[n, i] / 4;
                    }

                }

                Console.WriteLine("Presione Enter para iniciar entrenamiento
Comando 1");
                Console.ReadKey();
```

```
                for (int n = 0; n < entrena; n++)
                {
                    p.Run(O1_RAW, O2_RAW, P7_RAW, P8_RAW, n);
                    System.Threading.Thread.Sleep(100);


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

                        Entrena_Comando1[n,i] = (O1_RAW[n, i] + O2_RAW[n, i] +
P7_RAW[n, i] + P8_RAW[n, i])/4;

                    }

                }

                Console.WriteLine("Presione Enter para iniciar entrenamiento
Comando 4");
                Console.ReadKey();

                for (int n = 0; n < entrena; n++)
                {
                    p.Run(O1_RAW, O2_RAW, P7_RAW, P8_RAW, n);
                    System.Threading.Thread.Sleep(100);


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

                        Entrena_Comando4[n, i] = (O1_RAW[n, i] + O2_RAW[n, i]
+ P7_RAW[n, i] + P8_RAW[n, i])/4;

                    }

                }

////////////////////////////////////////////////////////////////////////////


                //////// /////////TRANSFORMADA DE FOURIER DE MODO NEUTRAL
//////////////////
                for (int n = 0; n < entrena; n++)
                {

                  TTF_EEG(O1_Neutral, n);
                  TTF_EEG(O2_Neutral, n);
                  TTF_EEG(P7_Neutral, n);
                  TTF_EEG(P8_Neutral, n);
                  TTF_EEG(SUMA_Neutral, n);
                  TTF_EEG(PROMEDIO_Neutral, n);
                  TTF_EEG(Entrena_Comando1, n);
                  TTF_EEG(Entrena_Comando4, n);

                 }


                //////////////////////      LECTURA DE ESTIMULOS
//////////////////////////

                    Console.WriteLine("Presione Enter para inicio de
lectura");
```

```
                    Console.ReadKey();

                     while(true)
            {
                    p.Run(O1_RAW,O2_RAW,P7_RAW,P8_RAW,0);
                          System.Threading.Thread.Sleep(1000);

                for(int n=0; n<entrena; n++)
                {
                    for (int i = 0; i < muestra; i++)
                    {

                        SUMA_RAW[n, i] = O1_RAW[n, i] + O2_RAW[n, i] +
P7_RAW[n, i] + P8_RAW[n, i];
                        PROMEDIO_RAW[n, i] = SUMA_RAW[n, i] / 4;
                    }
                }


////////////////////////////////////////////////////////////////////////////
                    ////////////////////////   ANALISIS DE FOURIER
/////////////////////////////


                    /////////// TRANSFORMADA DE FOURIER DE MODO ESTIMULO
//////////////////////

                    for (int n = 0; n < entrena; n++)
                    {
                        TTF_EEG(O1_RAW, n);
                        TTF_EEG(O2_RAW, n);
                        TTF_EEG(P7_RAW, n);
                        TTF_EEG(P8_RAW, n);
                        TTF_EEG(SUMA_RAW, n);
                        TTF_EEG(PROMEDIO_RAW,n);
                    }


////////////////////////////////////////////////////////////////////////////

                    ////////////////////////// COMPARACION
/////////////////////////////////////

                    for (int n = 0; n < entrena; n++)
                    {

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

                            PROMEDIO_absoluto[i] = PROMEDIO_RAW[n, i] -
Entrena_Comando1[n, i];

                        }
                        Absoluto1 = 0;

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

                            Absoluto1 = Absoluto1 + PROMEDIO_absoluto[i];

                        }
```

```
                        Console.WriteLine("El valor de absoluto Comando 1 es"
+ Absoluto1);

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

                            PROMEDIO_absoluto[i] = PROMEDIO_RAW[n, i] -
Entrena_Comando4[n, i];

                        }
                        Absoluto4 = 0;


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

                            Absoluto4 = Absoluto4 + PROMEDIO_absoluto[i];

                        }
                        Console.WriteLine("El valor de absoluto Comando 4 es"
+ Absoluto4);



                        for (int i = 0; i < muestra; i++)
                        {
                            absoluto_O1[i] = O1_RAW[n, i] - O1_Neutral[n, i];
                            absoluto_O2[i] = O2_RAW[n, i] - O2_Neutral[n, i];
                            absoluto_P7[i] = P7_RAW[n, i] - P7_Neutral[n, i];
                            absoluto_P8[i] = P8_RAW[n, i] - P8_Neutral[n, i];
                            SUMA_absoluto[i] = SUMA_RAW[n, i] -
SUMA_Neutral[n, i];
                            PROMEDIO_absoluto[i] = PROMEDIO_RAW[n, i] -
PROMEDIO_Neutral[n, i];
                        }


                        ///// FILTRO PASABANDA ///
                    //    System.Console.WriteLine("Electrodo O1");
                      // filtro_pasabanda(absoluto_O1);
                        //System.Console.WriteLine("Electrodo O2");
                        //filtro_pasabanda(absoluto_O2);
                        //System.Console.WriteLine("Electrodo P7");
                        //filtro_pasabanda(absoluto_P7);
                        //System.Console.WriteLine("Electrodo P8");
                        //filtro_pasabanda(absoluto_P8);
                        //System.Console.WriteLine("Suma de electrodos");
                        filtro_pasabanda(SUMA_absoluto);
                        System.Console.WriteLine("Promedio de electrodos");
                        filtro_pasabanda(PROMEDIO_absoluto);

                     //  brazo_robot(Comando,Jaco_m);

                        Comunicar(Comando);


                    }
                    // Application.SetCompatibleTextRenderingDefault(false);
                   // Application.Run(new Form1(SUMA_absoluto, muestra));
                  //  Console.ReadKey();
```

```
                }


            }//end while
            Console.WriteLine("End program");
        }

          Program()
        {
            engine = EmoEngine.Instance;
            engine.UserAdded += new
EmoEngine.UserAddedEventHandler(engine_UserAdded_Event);
            engine.Connect();
        }

        void engine_UserAdded_Event(object sender, EmoEngineEventArgs e)
        {
            Console.WriteLine("User Added Event has occured");
            userID = (int)e.userId;

            engine.DataAcquisitionEnable((uint)userID, true);
            engine.EE_DataSetBufferSizeInSec(1);

        }//end engine_UserAdded_Event


        void Run(double[,] o1, double[,] o2, double[,] p7, double[,] p8, int
n)
        {



            engine.ProcessEvents();




            if ((int)userID == -1)
                return  ;

            Dictionary<EdkDll.EE_DataChannel_t, double[]> data =
engine.GetData((uint)userID);
            if (data == null)
            {
                return;
            }//end if

            int _bufferSize = data[EdkDll.EE_DataChannel_t.TIMESTAMP].Length;

            Console.WriteLine("Writing " + _bufferSize + " lines of data ");


            for (int i = 0; i < _bufferSize; i++)
            {
                foreach (EdkDll.EE_DataChannel_t channel in data.Keys)
                {

                    if (channel == EdkDll.EE_DataChannel_t.O1)
                    {
                        o1[n, i] = Math.Round(data[channel][i], 2);
```

```
                    }
                    if (channel == EdkDll.EE_DataChannel_t.O2)
                    {
                        o2[n, i] = Math.Round(data[channel][i], 2);

                    }

                    if (channel == EdkDll.EE_DataChannel_t.P7)
                    {
                        p7[n, i] = Math.Round(data[channel][i], 2);

                    }

                    if (channel == EdkDll.EE_DataChannel_t.P8)
                    {
                        p8[n, i] = Math.Round(data[channel][i], 2);

                    }
                }

            }
            //end for

            //            EEG_data = GyroX + "," + GyroY + "," + F3 + "," + F4
+ "," + AF3 + "," + AF4;
            //EEG_data = GyroX + "," + GyroY + "," + F3 + "," + F4 + "," +
AF4;

            //for (int i = 0; i < _bufferSize; i++)
            //{
            //    foreach (EdkDll.EE_DataChannel_t channel in data.Keys)
            //        EEG_data = EEG_data+data[channel][i] + ",";
            //}//end for

           // EEG_data = EEG_data + "\0";
            // Console.WriteLine(EEG_data);
            // SendUDPData(IP, 27250, EEG_data);

            // if (blink < 3750)
            //    Console.WriteLine("----> blink");
          //  for (int i = 0; i < 64; i++)
          //  {
         //       aux[0, i] = EEG_Neutral[0, i];
          //  }

        }//end Run


        static void filtro_pasabanda(double[] absoluto)
        {

            double ganancia1 = 10;
            double ganancia2 = 10;
            double ganancia3 = 10;
            double ganancia4 = 10;

            for (int i = 0; i < 16; i++)
            {
                absoluto[i] = absoluto[i] * 0;
            }

            absoluto[20] = 0;
```

```
                absoluto[24] = 0;
                absoluto[28] = 0;

                for (int i = 17; i < 20; i++)  // 12.02 Hz
                {
                    absoluto[i] = absoluto[i] * ganancia1;
                }

                for (int i = 21; i < 24; i++) //14.69Hz
                {
                    absoluto[i] = absoluto[i] * ganancia2;
                }

                for (int i = 25; i < 28; i++) //17.36Hz
                {
                    absoluto[i] = absoluto[i] * ganancia3;
                }

                for (int i = 29; i < 32; i++)//20.03Hz
                {
                    absoluto[i] = absoluto[i] * ganancia4;
                }

                int max = 1;

                for (int i = 0; i < 32; i++)
                {
                    if (absoluto[i] > absoluto[max])
                    {
                        max = i;
                    }

                }

                if (absoluto[max] >= umbral)
                {
                    Console.WriteLine("Frecuencia detectada: " + max * 1.0016 +
"Hz" + " Valor : " + absoluto[max]);
                    if (max > 28 && max < 32)
                    {
                        Console.WriteLine("COMANDO 4");
                        Comando = 4;
                    }

                    if (max > 24 && max < 28)
                    {
                        Console.WriteLine("COMANDO 3");
                        Comando = 3;
                    }

                    if (max > 20 && max < 24)
                    {
                        Console.WriteLine("COMANDO 2");
                        Comando = 2;
                    }

                    if (max > 16 && max < 20)
                    {
                        Console.WriteLine("COMANDO 1");
                        Comando = 1;
                    }
```

```
            }

            else
            {
                Console.WriteLine("COMANDO 0");
                Comando = 9;
            }

        }

        static void TTF_EEG(double[,] EEG_data, int n)
        {

            System.Numerics.Complex[] complejo = new
System.Numerics.Complex[muestra];

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

                complejo[i] = new System.Numerics.Complex(EEG_data[0, i], 0);
            }

            MathNet.Numerics.IntegralTransforms.Fourier.Forward(complejo);


            EEG_data[n,0] = 1;

            for (int i = 1; i < muestra; i++)
            {
                EEG_data[n,i] = System.Numerics.Complex.Abs(complejo[i]);


            }


        }

        static void brazo_robot(int comando, Kinova.API.Jaco.CJacoArm Jaco_m)
        {


            Jaco_m.ControlManager.StartControlAPI();

            //Declare and initialize a CJoystickValue that will emulate a
GeneralJoystick



            Kinova.DLL.Data.Jaco.CJoystickValue value = new
Kinova.DLL.Data.Jaco.CJoystickValue();


            if (Jaco_m.JacoIsReady())
            {



                // int mensaje = serial_comm();
                int mensaje = comando;


                Console.WriteLine(mensaje);
```

```
if (mensaje != 255)
{

    Jaco_m.ControlManager.StartControlAPI();
    // X UP
    // if (((Button)m_Grid.Children[4]).IsPressed)
    if (mensaje == 4)
    {
        value.InclineLR = -1f;
    }

    // Y UP
    //if (((Button)m_Grid.Children[5]).IsPressed)
    if (mensaje == 3)
    {
        value.InclineFB = -1f;
    }

    // X DOWN
    //if (((Button)m_Grid.Children[7]).IsPressed)
    if (mensaje == 2)
    {
        value.InclineLR = 1f;
    }


    // Y DOWN
    //if (((Button)m_Grid.Children[8]).IsPressed)
    if (mensaje == 1)
    {
        value.InclineFB = 1f;
    }

    // Z DOWN
    if (mensaje == 9)
    {
        value.Rotate = 1f;

    }

    // Z UP
    if (mensaje == 8)
    {
        value.Rotate = -1f;

    }

    //abre grip
    //   if (mensaje == 6)


    //abre grip
    // if (mensaje == 7)

    // Store Position
    if (mensaje == 5)
    {

        value.ButtonValue[2] = 1;
    }
    /*
    // Go To Position
```

```
                    if (((Button)m_Grid.Children[11]).IsPressed)
                    {
                        value.ButtonValue[7] = 1;
                    }*/

                    //We update JACO with the new data.


////////////////////////////////////////////////////////////
                    Jaco_m.ControlManager.SendJoystickFunctionality(value);


                    if (mensaje == 15)
                    {

                        value.InclineFB = 0;
                        value.InclineLR = 0;
                        value.Rotate = 0;

                        Thread.Sleep(50);

Jaco_m.ControlManager.SendJoystickFunctionality(value);
                    }
                }
            }
        }


        static void Comunicar(int comando)
        {
            //while (true)
            // {
                Socket cliente = new Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);
                IPEndPoint MiDireccion = new
IPEndPoint(IPAddress.Parse("192.168.1.33"), 1234);

                //   try

                cliente.Connect(MiDireccion);
                Console.WriteLine("Conectado con exito,ingrese mensaje:");
                string msg = comando.ToString();
             //string msg = Console.ReadLine();
                byte[] msgBuffer = Encoding.Default.GetBytes(msg);
                cliente.Send(msgBuffer, 0, msgBuffer.Length, 0);
                Thread.Sleep(100);

                byte[] buffer = new byte[255];
                int rec = cliente.Receive(buffer, 0, buffer.Length, 0);
                Array.Resize(ref buffer, rec);

                Console.WriteLine("Recibido: {0}",
Encoding.Default.GetString(buffer));
                cliente.Close();
          //  }

        }


    }

}
```

Programa Arduino

```
void setup() {
  // put your setup code here, to run once:
pinMode(13,OUTPUT);
pinMode(12, OUTPUT);
pinMode(8, OUTPUT);
pinMode(7, OUTPUT);
#define periodo1 1000/31
#define periodo2 1000/30
#define periodo3 1000/29
#define periodo4 1000/28
}

void loop() {
  // put your main code here, to run repeated

int cont1=0;
int cont2=0;
int cont3=0;
int cont4=0;

 while(true)
{
  delay(1);
  cont1=cont1+1;
  cont2=cont2+1;
  cont3=cont3+1;
  cont4=cont4+1;
// LED 1 //
  if(cont1==(periodo1/2))
    {
      digitalWrite(13, HIGH);
    }
  if(cont1==periodo1)
   {
      digitalWrite(13, LOW);
      cont1=0;
   }

// LED 2 //

  if(cont2==(periodo2/2))
  {
      digitalWrite(12, HIGH);
  }
  if(cont2==periodo2)
   {
      digitalWrite(12, LOW);
      cont2=0;
   }

   // LED 3 //
```

```
  if(cont3==(periodo3/2))
  {
      digitalWrite(8, HIGH);
  }
  if(cont3==periodo3)
   {
      digitalWrite(8, LOW);
      cont3=0;
   }

   // LED 4 //

  if(cont4==(periodo4/2))
  {
      digitalWrite(7, HIGH);
  }
  if(cont4==periodo4)
   {
      digitalWrite(7, LOW);
      cont4=0;
   }
}

}
```