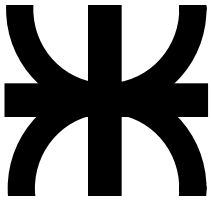


UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL SAN NICOLÁS
INGENIERIA EN ELECTRÓNICA

TÉCNICAS DIGITALES III

Problema de ingeniería

SISTEMA "PLAYER TRACKING" PARA MAQUINAS TRAGAMONEDAS

Integrantes:
- Famá, Angel

Docentes:
- Profesor: Ing. Poblete, Felipe
- Auxiliar: Ing. González, Mariano

AÑO 2017

Tabla de contenido

1. Introducción

En este informe, exponemos la planificación, diseño y ejecución de un desarrollo para la resolución de un problema de ingeniería de aplicación real. En el marco de la catedra de Técnicas Digitales 3, se pretende aplicar los temas desarrollados en las clases durante el año a una solución real, técnica y comercialmente viable.

El problema planteado incurre en la necesidad de un desarrollo nacional, de bajo costo con funciones personalizadas de sistemas interconectados de datos en máquinas tragamonedas con el fin de optimizar los servicios en salas de apuestas como casinos y bingos. En particular el sistema al que hacemos referencia en este trabajo es el de PLAYER TRACKING o seguimiento de clientes, considerando el mismo como base de entrada en un mercado con gran potencialidad de desarrollos.

La solución que aquí es expuesta, es parte inicial de un proyecto más abarcativo cuya necesidad surge de un potencial cliente del sector privado: Bingo Ramallo S.A. Esta sala de apuestas ubicada en la localidad de Ramallo, es una de las 46 salas de bingo de la provincia de Buenos Aires que están habilitadas por el I.P.LyC. (Instituto Provincial de Loterias y Casinos) para explotar Máquinas Electrónicas de juegos de azar o EGM (Electronic Gaming Machine).

Bingo Ramallo S.A. con un total de 405 terminales de juego instaladas y un plantel de personal directo de más de 250 empleados, juega un papel muy importante en la economía del municipio ya que además de brindar las fuentes de trabajo mencionadas, atrae turistas y visitantes de localidades vecinas que enriquecen los emprendimientos turísticos que fomenta la ciudad.

Actualmente la licencia de la sala de bingo número 22, es propiedad de la Biblioteca Popular José Manuel Estrada de la municipalidad de Ramallo que contrata a Bingo Ramallo S.A. para su explotación, siendo el ente público participe de las utilidades. Por otro lado, el IPLyC es socio de la explotación recibiendo un 34% de las utilidades brutas. Los ingresos de Lotería de la Provincia por este canon representan $20.000 millones anuales, de los cuales, 15% los retiene el organismo para su financiamiento y el resto se distribuye a los distintos ministerios y programas sociales.

Si bien el negocio de las salas de bingos y casinos genera cierta apatía en parte de la sociedad y además, son de público conocimiento los altos márgenes de rentabilidad que las empresas obtienen de la explotación de este negocio a costa, en muchos casos, del empobrecimientos del propio cliente, la regulación responsable por parte del estado y la posibilidad de aplicación de tecnologías y desarrollos nacionales, terminan redundando en mayores beneficios sociales en comparación con estados donde el juego se prohíbe y prolifera la ilegalidad. Por tal motivo, consideramos que los sistemas que aporten a la trasparencia y legalidad del juego, son beneficiosos tanto para las empresas como para la entera sociedad.

1.1 ¿Qué es Player tracking en máquinas tragamonedas?

La idea del Player Tracking en los tragamonedas puede atribuirse en sus comienzos a una promoción lanzada por el casino Harrah's en Las vegas, las misma se llamaba "Premium Points". Inicialmente en los 70s la promoción funcionaba de la siguiente forma, un empleado del casino, observaba a los jugadores en determinados tragamonedas y realizaban un cupón al a los mismos por cada 20 dólares jugados, estos cupones podían acumularse y ser cambiados por premios dentro del propio casino como electrodomésticos u ofertas de gastronomía. El objetivo primario en sus comienzos, era premiar al cliente fiel y conseguir asi que el mismo permanezca más tiempo en el casino.

Manteniendo este concepto, con los sistemas actuales, la mayoría de los grandes casinos utilizan la herramienta del player tracking ajustando políticas comerciales que, aplicadas desde el marketing estratégico, permite atraer, conocer, segmentar y fidelizar a los clientes. Los objetivos principales del mismo se pueden dividir en tres destinatarios:

- El cliente: Obtener un reconocimiento por parte de la sala por su fidelidad como cliente, obtener acceso de manera automática a salas VIP, participar de promociones y beneficios como cliente activo.

- El operador: Obtener estadística e información fehaciente de los volúmenes de juegos de sus clientes en fechas y horarios, posibilitando la programación de concursos y promociones, permitiendo además complementar las estadísticas propias de las maquinas con los gustos particulares de clientes habituales.

- El ente regulador: El acceso a la información, podría permitir controlar eventuales fraudes, comportamientos compulsivos o incluso, prevenir el lavado de dinero.

Esta herramienta puede formar parte de un sistema de gestión más amplio en la sala como un CASHLESS o T.I.T.O. o bien ser independiente de los mismos, en tal caso pasaría a ser un sistema adicional y puede dividirse en dos grandes partes:

- Sección de campo: consta del hardware a instalar en cada máquina de la sala que identificara al apostador y reportara las transacciones a la sección superior. Para identificar al cliente son utilizadas tarjetas magnéticas, smart card o RFID instalados en cada tragamonedas, además se dispone de un display que brinda información al cliente como resumen de cuenta e incluso comunica si el mismo es acreedor de algún premio o beneficio y en algunos se incluye un teclado que permite interactuar al cliente o resguardar los datos de la tarjeta mediante la implementación de un pin. Este dispositivo en forma de panel, se monta en la maquina tragamonedas y se conecta a la misma mediante la utilización de un puerto serie con protocolo SAS® (SLOT ACOUNTING SYSTEM) que le permite obtener la información de los créditos apostados. Además se vincula con la sección de sistema, mediante una conexión que puede ser CAN o Ethernet y un protocolo que es

elegido por el fabricante del sistema. En la figura 1.1 se puede observar esta sección integrada en los tragamonedas.

Figura 1.1 _ Paneles Player Tracking

- Sección de sistema: consta del hardware y software necesario para tres funciones bien definidas, en principio la administración de los clientes, alta, baja y generación de las tarjeta, la administración y gestión de las terminales de canje de premios, generalmente en ubicadas en las cajas de cambio y luego la recolección y almacenamiento en base de datos de la información y transacciones de la sección de campo

1.2 ¿Que se ofrece hoy en el mercado?

Los fabricantes internacionales de tragamonedas como IGT, ARISTOCRAT o SG Gaming (https://www.sggaming.com/Systems/Player-Tracking) y otros proveedores de sistemas nacionales ofrecen dentro de sus carteras de productos soluciones con estas funcionalidades como una herramienta adicional a sistemas más completos. No obstante, los mismos están pensados para salas de jugo de gran envergadura con múltiples locaciones y funciones avanzadas como por ejemplo: torneos entre cliente, interconectividad entre diferentes salas, premios sorpresa, participación de sorteos on-line, soluciones cardless, club de jugadores, control de ingreso, etc. Esto termina haciéndolos costos e inviables para la implementación en muchas salas de juego pequeñas (menos de 500 posiciones de juego) como las que se encuentran habitualmente en el interior en nuestro país. Generalmente estos sistemas no se venden al operador de sala, el negocio se realiza mediante contratos de explotación conjunta, es decir, el operador abona un porcentaje de la recaudación a proveedor de sistema y además se hace cargo de los costos de implementación.

1.3 ¿Cuál es la diferencia de nuestra solución?

La solución que planteamos, es un desarrollo de bajo costo de hardware, por ende de un menor valor de implementación y además un software depurado de las funcionalidades extras y limitado a las funciones esenciales del sistema.

Nuestro sistema se limita a fidelizar los clientes y contabilizar las apuestas del mismo. Esto obviamente tiene sus desventajas en cuanto funcionalidad, pero no debemos perder de viste el principal objetivo que es obtener información fehaciente de los clientes, mediante una inversión limitada.

Por otro lado, si bien aún se encuentran en procesos de reglamentación, es inminente la aplicación en la provincia de Bs. As. el cobro de entradas a las salas de juego mediante tarjeta como las utilizadas en el sistema de player traking. Si la sala requiere un sistema que cumpla solamente esta funcionalidad, para cumplir la normativa, no tendría sentido invertir en un sistema enfocado a otro objetivo como son los ofrecidos actualmente en el mercado. Este es el nicho que nuestro sistema trata de aprovechar. En la siguiente nota (salvando algunas aberraciones técnicas), describe esta nueva imposición por parte del estado: http://www.lanacion.com.ar/2048807-una-tarjeta-para-pagar-la-entrada-a-los-bingos-servira-para-el-control-de-ludopatas

1.4 Componentes y descripción del sistema completo, puntos a resolver.

El sistema completo se puede dividir en las siguiente cuatro grandes parte a resolver:

- Panel Player Tracking (PPT): Es la parte fundamental de la sección de campo, básicamente será la interface entre el cliente, la EGM y el sistema. En el mismo, el cliente inserta su tarjeta identificatoria y observa en el display la información que el sistema le brinda. La EGM se conecta con el mismo para informar las apuestas realizadas. Por ultimo transmite la información al servidor central.

- Servidor central: Es la parte fundamental de la sección de sistema, el mismo se encarga de recolectar la información recibida de los PPT y almacenar los registros en una base de datos. Además provee acceso a dicha base de datos, a las aplicaciones de gestión y administración.

- Infraestructura de conexión: Es el medio mediante el cual se comunican los PPT y el servidor, incluyendo los dispositivos de interconexión.

- Aplicaciones de gestión de clientes: También forma parte de la sección de sistema, contempla las aplicaciones finales que utilizará el operador de la sala para obtener la información desde la base de datos del servidor central. Además de leer información, mediante estas aplicaciones se realizan las cargas de clientes, generación de tarjetas, habilitaciones, canjes de premios etc.

El funcionamiento del sistema comienza mediante el alta de un cliente en una terminal designada para tal fin, previo firma de condiciones se le asigna al cliente una tarjeta que lo identificara dentro del sistema. Cuando el cliente decide realizar una apuesta en una maquina determinada, introduce en el PPT su tarjeta realizando un proceso de login y desde ese momento, se le acreditaran puntos promocionales dependiendo de los montos apostados, el logout se realiza automáticamente al quitar la tarjeta del panel. Estos puntos acumulados podrán ser canjeados en los terminales instalados por premios o beneficios que disponga la sala. Adicionalmente se podrán realizar sorteos con los clientes logueados en el sistema mostrando los premios e indicaciones en el display del panel. Mediante esto, la sala recolectara información de

apuestas de cada cliente dado de alta en el sistema y en las terminales de gestión, se procesa esa información. Cabe destacar que en todas las salas este sistema es opcional al cliente y no excluye al mismo del resto de los servicios, pudiendo realizar apuestas sin necesidad de tener tarjeta de PT. Pero, en caso de proliferar las restricciones que impulsa la provincia de Bs. As., será condición necesaria para poder jugar en las EGM.

En el esquema de la figura 1.2 podemos ver la interconexión de los cuatro puntos descriptos y es el esquema que consideraremos para el desarrollo de nuestro sistema. En algunos casos, pueden ofrecerse otro tipo de distribuciones, sobre todo cuando Player Traking es para integral de un sistema más complejo.

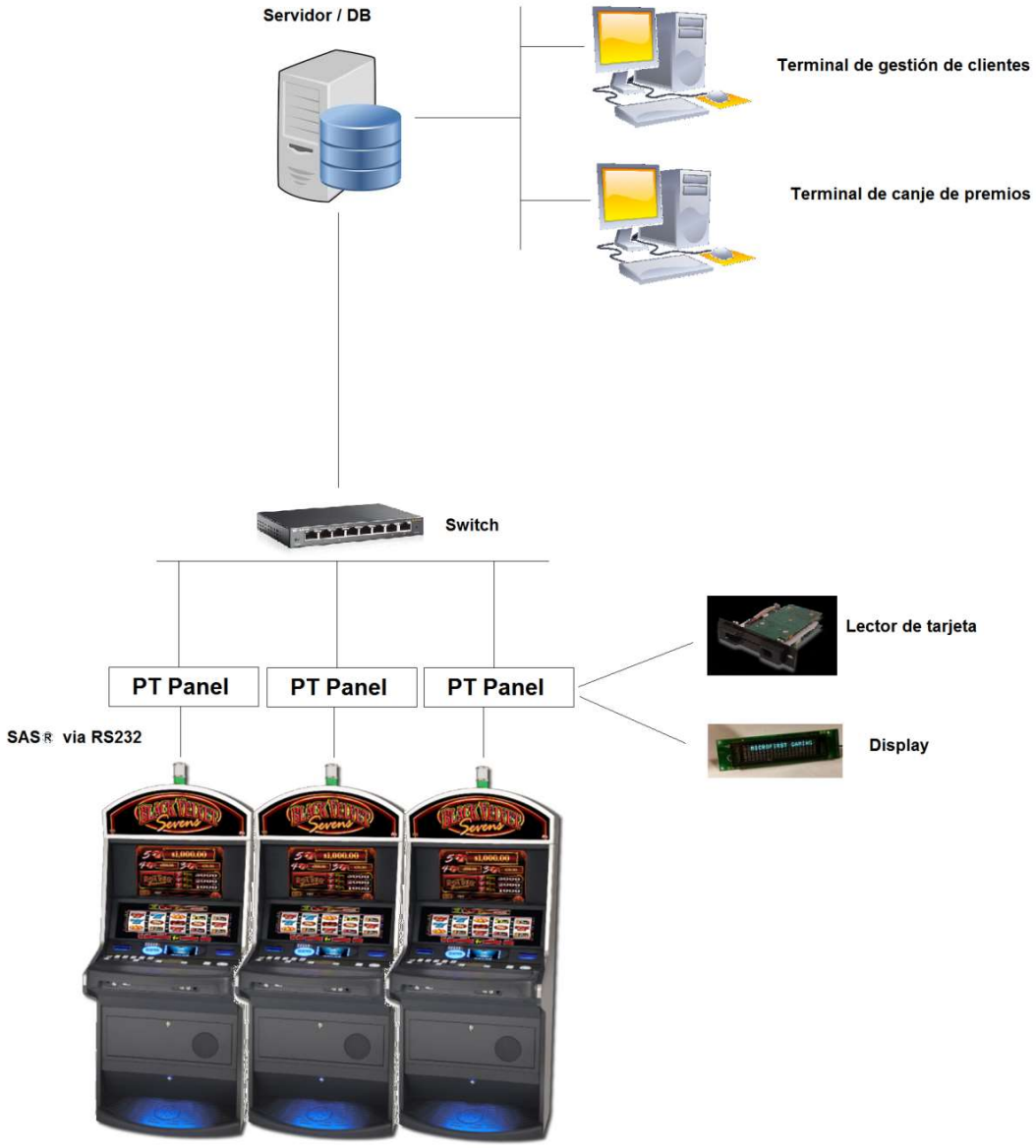


Figura 1.2 _ Sistema Player Tracking

1.5 ¿Cuál es el alcance que establecimos en este trabajo?

El proyecto completo consta de un gran tamaño en cuanto a desarrollo, sobre todo en lo que a software respecta. Si bien el objetivo final es poder llevarlo a cabo en un 100%, con lo cual no podemos subestimar el diseño, escaparía a los tiempos establecidos por el curso de la materia llegar a resolver todas las instancias que lo componen. En tal sentido, proponemos acotar los márgenes del alcance y concentrar los esfuerzos en una sección particular, pero central del proyecto.

Para ser preciso la propuesta consiste en desarrollar los siguientes puntos:

- Determinar el hardware a implementar en el panel.
- Diseño de una red LAN cableada como medio de conexión entre paneles y servidor para 500 terminales, elección de materiales y equipamiento.
- Diseño e implementación de un protocolo de aplicación adecuado para la comunicación bidireccional TPC/IP que permita: Del panel a al servidor reportar el login y logout del cliente incluyendo la identificación del slot donde se realiza y la cantidad de apuestas realizadas. Desde el servidor al panel la posibilidad de mostrar información en el display del mismo como la identificación del cliente y la acreditación de un premio.
- Programación del hardware del panel contemplando la comunicación con el servidor con posibilidad de gestión de mantenimiento.
- Programación en alto nivel del servidor de comunicaciones que responda las solicitudes de los paneles y almacene los datos en un archivo.

Se simularan mediante simplificaciones u omitirán las siguientes instancias:

- La comunicaciones del panel con la maquina tragamonedas. En su lugar ser reportara una apuesta al panel, mediante dos botones con diferentes valores apostados.
- El lector de tarjetas se simulara mediante una pequeña aplicación que por el puerto serie de una pc permitirá enviar el número de tarjeta al hardware del panel.
- No se desarrollaran las aplicaciones finales de análisis de datos y gestión de los mismos.
- No se determinara ni dimensionara el hardware necesario para el servidor, pero propondremos un estándar de uso común en estos sistemas.
- No se probará el sistema con múltiples conexiones simultaneas, por lo que quedara pendiente evaluar la respuesta del servidor en condicione de uso real.

2. Desarrollo

2.1 Funcionamiento del sistema a desarrollar en este trabajo

Como se explicó en la sección anterior, se desarrollaran las funciones básicas para garantizar la comunicación panel, basados en este fin explicaremos cual será el funcionamiento que esperamos conseguir al finalizar este trabajo.

La comunicación entre el panel y el servidor estará basada en TCP/IP bajo una arquitectura de cliente/servidor. El cliente será el Panel, que ante la necesidad de reportar un evento, enviara una petición al servidor y este responderá inmediatamente con la información requerida. Los eventos comunicables serán:

- Sincronización y registro en el startup del panel
- Login del cliente (ingreso de tarjeta)
- Logut del cliente (extracción de la tarjeta)
- Apuesta realizada
- Señal de vida o Keep-alive para mantenimiento de conexión

Cuando un PPT se encienda o reinicie, el primer evento que comunicara será el registro del panel en el servidor, para eso enviara la solicitud adecuada con el número de identificación del mismo, la identificación del terminal al que se encuentra conectados y un reporte de estado, en caso de obtener respuesta del servidor, la misma constara de la confirmación del registro junto con la referencia de fecha y hora para que se registre en el panel. Cuando la comunicación falle en este punto, el panel quedará desconectado del sistema y no podrá ofrecer ninguna otra funcionalidad al cliente hasta que sea subsanado el inconveniente.

La inserción o extracción de tarjetas, se simulará mediante una aplicación que utilizara un puerto USB. El hardware que se pretende utilizar para resolver este punto es un lector de RFID como el RC552 que se comunica por un puerto serie, es por ello que en primer mediada la simulación mediante una aplicación por puerto serie es adecuada para esta etapa de desarrollo.

Cuando se realiza una inserción, por el puerto serie, el micro del panel recibe el número de tarjeta. Con este dato arma el evento de longin que envía al servidor. Cuando el servidor recibe el evento responde al panel con la confirmación del login y los datos requeridos. Si el login es exitoso, se mostraran en el panel el número de identificación del cliente y los créditos actuales del mismo. Si el servidor no responde el evento, el login no se realiza y el panel muestra un error solicitando un nuevo intento.

Una vez que el cliente se encuentre logueado en el panel, cada vez que realice una apuesta, se informara el evento al servidor para que se registrará el evento y responderá con una confirmación de recepción y con el total de puntos del cliente que se actualizara en el display. En caso de que una apuesta no sea registrada por el servidor por ejemplo, por un problema de comunicación, permanecerá almacenada en la memoria del panel hasta que efectivice el registro, si el cliente sigue apostando, se sumaran todas las apuestas hasta tanto el inconveniente de comunicación sea subsanado

y se reporte al servidor. La realización de apuestas se simulara mediante dos botones con diferentes importes a modo demostrativo, el desarrollo final recibirá esta información de la EGM.

Cuando se realice la extracción de la tarjeta, también mediante el software de simulación de tarjeta, se enviara el evento al servidor reportando esta condición, el mismo responderá con una confirmación y el panel dará por terminada la sección limpiando los valores de las variables del cliente particular.

Para garantizar la conectividad entre paneles y servidor cuando no se esté realizando apuestas, o no haya eventos en un intervalo determinado de tiempo, el panel enviara un evento de keep alive o señal de vida al servidor que responderá en caso de recibirlo con un "acknowledge". En caso de no recibir el ACK, se indicara en el panel esta situación para permitir la resolución del inconveniente.

El panel tendrá implementado un servidor http el cual se podrá acceder mediante cualquier terminal que se encuentre en la red, dicho servicio se utilizara para mantenimiento, mostrando estado actual o errores en el panel.

Si bien, en esta etapa las funcionalidades son limitadas, se abarcan todos los puntos previstos en el alcance que se le da a este trabajo siendo cada una de ellas, bases suficientes como para establecer las líneas de desarrollo del proyecto en su totalidad.

.

2.2 Evaluación de la comunicación y flujo de información

Para determinar el volumen de información que requerirá transmitir y recibir cada panel, estudiamos en la sala de Bingo Ramallo S.A. las estadísticas de juego de algunas EGM más relevantes, en promedio pudimos determinar que entre apuesta y apuesta como mínimo hay un tiempo de 2 segundos, el mismo es el que requiere la animación de los rodillos desde que se presiona el botón de juego hasta que muestra el resultado. Dependiendo del cliente, este tiempo puede alargarse si decide jugar pausadamente, por el contrario si se activa el modo de juego automático donde la maquina inicia una nueva jugada tras otra, los tiempo rondan entre 2 y 3 segundo dependiendo de la maquina en particular.

Por otro lado, consultado con personal de sala y de monitoreo, el recambio de jugadores en cada máquina es muy variable, desde los 5 minutos que pueden tardar un cliente en realizar unas 20 jugadas, hasta algunos casos donde el cliente no se levanta de la EGM por hasta 4 o 5 horas. Definimos a los efectos de calcular la cantidad de login y logout un tiempo medio de juego por EGM de 15minutos.

Con esto dos valores, podemos determinar cuál será el máximo intercambio de información entre los paneles y el servidor suponiendo un total de 500 terminales:

$$F_{BET} = \frac{500}{2} = 250 \ \frac{EVENTOS}{SEG} \text{ Frecuencia de apuestas}$$

$$F_{LOG} = \frac{500}{15 \cdot 60} = 0.57 \ \frac{EVENTOS}{SEG} \text{ Frecuencia de logueos}$$

Como podemos ver, el servidor deberá poder atender como máximo, para 500 terminales, aproximadamente 250 peticiones por segundo para poder tener una respuesta en tiempo real. Si tenemos en cuenta, como veremos en el punto siguiente, que los datos que se trasmiten tienen un tamaño de 11 bytes y consideramos los 20 bytes que agrega el encabezado IP, como se puede ver en la siguiente figura 2.1, el datagrama para cada evento tendrá un tamaño de 31 bytes.

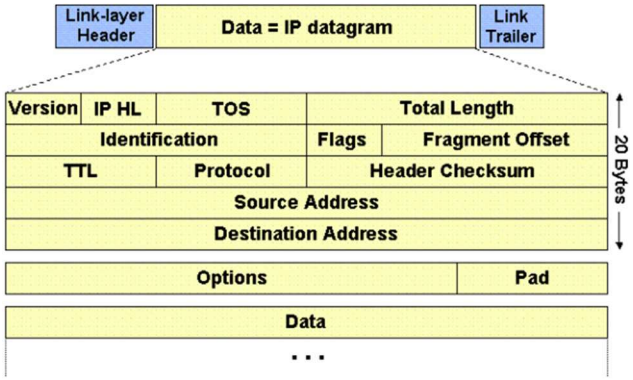


Figura 2.1

Podemos ahora estimar la velocidad que requerirá nuestra red para cumplir con los requerimientos del sistema. De los números anteriores y considerando que ante cada petición de un panel, el servidor responde con la misma cantidad de bytes:

$$\text{Flujo} = 250\text{eventos} \cdot 2 \cdot 31\text{bytes} \cdot 8\text{bits} = 121\text{kbps}$$

Como puede verse, el flujo de información requerido está muy por debajo incluso de un red LAN 10baseT de 10mbps, de todas formas, con las tecnologías actuales armar una red 10baseT no difiere en cuanto costos de una red 100baseT de 100mbps, con lo cual esta última opción es la elegida para resolver el problema.

Los cálculos anteriores, nos permiten tener la seguridad que la red no será un problema para la comunicación. La capacidad del servidor para atender las solicitudes y responder en tiempo real, será estudio de otro etapa, por el momento consideraremos que el mismo puede atender al totalidad de las mismas.

2.2 Dimensionamiento de la red LAN

Para el montaje de la red física, si tenemos en cuenta que la distribución de las EGM se realiza en islas en la sala, se instalaran switches en cada isla que convergerán a un swhitch colector. Para las 500 máquinas, serán necesarios al menos 22 switches de 24 bocas distribuidos en las distintas islas más el colector en el rack del servidor.

El cableado se realizara mediante cable FTP Cat5e, el mismo es requerimiento legal por parte del IPLyC para sistemas interconectados de máquinas tragamonedas, debido a la inmunidad al ruido que le ofrece el mallado sobre el cable UTP. Se estiman para una sala de 500 máquinas, un total de 6 bobinas de 305mts. Debido también a la exigencia mecánica que se requiere por la constante actualización de la distribución de las EGM en la sala, se elige un cable del multifilar con vaina apta para exterior.

Cabe destacar que de acuerdo la cantidad de host que se van a interconectar, será suficiente un direccionamiento IPv4 aunque se requerirá una máscara de red por lo menos /23 (255.255.254.0). Para comodidad definiremos una máscara /16 (255.255.0.0) que permite más de 65.000 host. Esto es importante ya que por defecto todos los adaptadores asignan una máscara /24 y se deberá cambiar la misma.

2.4 Desarrollo del protocolo de comunicación panel - servidor

Con el fin de poder mantener una comunicación eficiente entre los paneles y el servidor de forma tal de transmitir solamente lo necesario y amparándonos en que el protocolo de comunicación utilizado TCP\IP que nos ofrece solucionado las capas inferiores como transporte, enlace, red y acceso al medio, nos concentramos en determinar un "protocolo" o codificación de la información a nivel aplicación.

Así planteado, determinamos que para los 6 eventos que deseamos comunicar entre PPT y el servidor, son suficientes 11bytes. En la tabla que mostramos a continuación está el detalle de cómo ser organizan los arreglos que transmite el cliente y la respuesta que debería recibir del servidor.

Protocolo para Player Tracking												
EVENTO	HOST	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8	Byte9	Byte10
REGISTRO PANEL	PTP	Comando	ID Panel		Status	IP3	IP4	EGM Asset Number				S. EGM
		0x01	0xNN	0xNN	0xNN	0xNN	0xNN	0xNN	0xNN	0xNN	0xNN	0xNN
	SERVIDOR	Comando	ID Panel		Fecha			Hora			Status	
		0x81	0xNN	0xNN	Año	Mes	Día	Hora	minuto	Seg	0xNN	
LOGIN CLIENTE	PTP	Comando	ID Panel		ID Cliente			Status				
		0x02	0xNN	0xNN	0xNN	0xNN	0xNN	0xNN				
	SERVIDOR	Comando	ID Panel		ID Cliente			Créditos cliente				Status
		0x82	0xNN	0xNN	0xNN	0xNN	0xNN	0xNN	0xNN	0xNN	0xNN	0xNN
APUESTA REALIZADA	PTP	Comando	ID Panel		ID Cliente			Créditos apostados				Status
		0x03	0xNN	0xNN	0xNN	0xNN	0xNN	0xNN	0xNN	0xNN	0xNN	0xNN
	SERVIDOR	Comando	ID Panel		ID Cliente			Créditos cliente				Status
		0x83	0xNN	0xNN	0xNN	0xNN	0xNN	0xNN	0xNN	0xNN	0xNN	0xNN
LOGOUT CLIENTE	PTP	Comando	ID Panel		ID Cliente			Créditos cliente				Status
		0x04	0xNN	0xNN	0xNN	0xNN	0xNN	0xNN	0xNN	0xNN	0xNN	0xNN
	SERVIDOR	Comando	ID Panel		ID Cliente			Status				
		0x84	0xNN	0xNN	0xNN	0xNN	0xNN	0xNN				
KEEP ALIVE	PTP	Comando	ID Panel		Status							
		0x0A	0xNN	0xNN	0xNN							
	SERVIDOR	Comando	ID Panel		Status							
		0x8A	0xNN	0xNN	0xNN							

A continuación detallamos que significa cada dato transmitido

- Comando: 1byte que identifica el evento trasmitido.
- Id Panel: 2 byte con la identificación del PPT, rango de 0 a 65535.
- IP3, IP3: 2 bytes, últimos dos byte de la dirección IP del panel.
- Status: 1 byte, estado del host que transmite.
- EGM Asset Number: 4bytes, número de identificación de la EGM, n° serie.
- S.EGM: 1byte, estado de la conexión del panel con la EGM.
- Id cliente: 3 bytes para identificar a cada cliente 2^{24} =16.7 millones.
- Créditos Cliente: 4bytes, créditos actuales del cliente 2^{32}=4,3 mil millones máximo.
- Créditos apostados: 4bytes, créditos apostados en la jugada por el cliente.

2.5 Hardware y programación del panel

2.5.1 Selección del hardware

Para cumplir con los requerimientos del proyecto, e hardware requiere las siguientes funcionalidades:

- Conectividad Ethernet
- Al menos 2 puertos serie uno para el RFID otro para la conexión de la EGM
- Puerto I2C o SPI para el display
- I/O de entradas general

Seleccionamos en principio una placa de desarrollo Arduino, figura 2.2, basada en el microcontrolador ATMEGA2560 de Atmel (o Microchip tras la reciente adquisición), de las denominadas “Arduino Compatible” ya que es copia de la placa original fabricada por ARDUINO en Italia. Posee las mismas prestaciones pero de fabricación china y con componentes de menor calidad que para una etapa de desarrollo son más que suficientes.

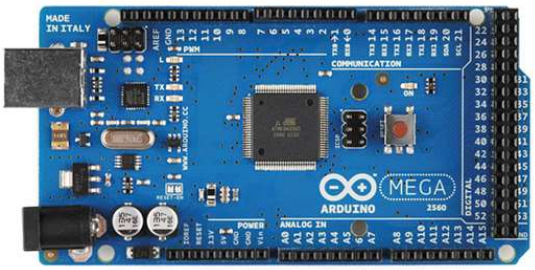


Figura 2.2

OVERVIEW | TECH SPECS | DOCUMENTATION

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

En cuanto al microcontrolador, las características más relevantes que lo determinaron apto para el proyecto son las siguientes:

- Microcontrolador de 8-bit con 256KBytes de memoria flash programable
- Atmel® AVR® con una avanzada arquitectura RISC
- Hasta 16 MIPS de rendimiento a 16MHz
- 4Kbytes EEPROM / 8Kbytes SRAM interna
- Cuatro puertos serie UART
- Cinco interfaces SPI
- Disponibilidad de bus I2C

Para más información se puede consultar la hoja de datos en el siguiente link: Data Sheet ATmega2560

Para agregarle a la placa de desarrollo la posibilidad de conexión Ethernet, utilizamos un shield del mismo fabricante que se conecta directamente sobre la misma, figura 2.3.

Figura 2.3

El controlador en el cual está basada la placa es el W5100, un controlador Ethernet 10/100 con todas las funciones en un solo chip diseñado para facilitar la tarea implementación de conectividad a Internet sin SO. El W5100 es compatible con los estándares IEEE 802.3 10BASE-T y con 802.3u 100BASE-TX. Incluye un Stack TCP / IP que admite TCP, UDP, IPv4, ICMP, ARP, IGMP y PPPoE. Posee un buffer interno de 16Kbytes para la transmisión de datos y soporta hasta 4 sokets simultáneos. Para la integración posee interfaces: acceso directo a memoria o SPI que es el usado en el shield.

Se eligió para mostrar la información al usuario un display de cristal líquido de 2x16 caracteres, ver figura 2.4, basado en el controlador Hitachi HD44780 con código CCM1620CSLS2, si bien el mismos se queda “corto” para el desarrollo definitivo, para esta etapa es suficiente. El mismo se conecta a 6 GPIO del micro, aunque en un futuro se pretende utilizar algo más avanzado como un OLED con conexión 12C. La hoja de datos puede verse en:
http://www.exa.unicen.edu.ar/catedras/microc/CCM1620CSL%20SPEC.pdf

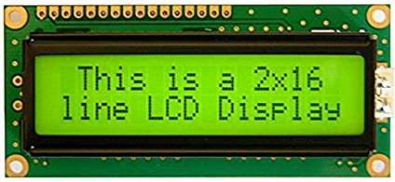


Figura 2.4

En el siguiente esquema, figura 2.5, se muestra el circuito e interconexión de los diferentes módulos. Para ver los circuitos completos de cada una de las placas de desarrollo referirse al fabricante www.arduino.cc :

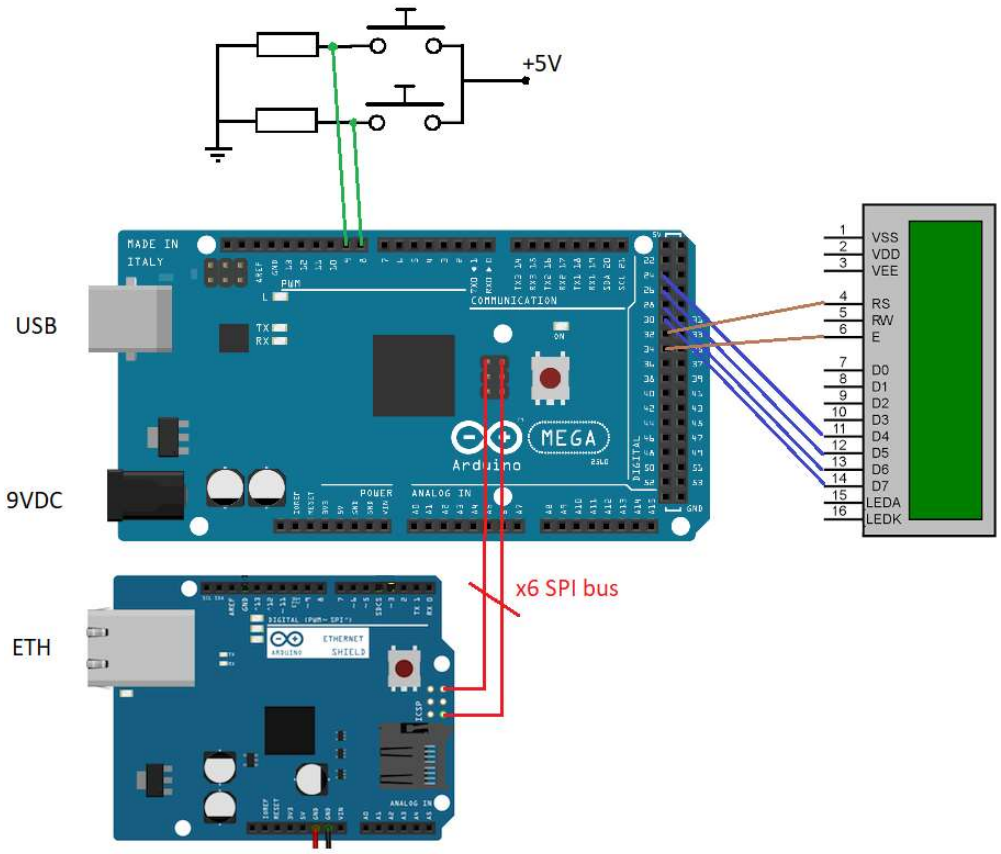


Figura 2.5

2.5.2 Programación del panel

La programación del microcontrolador se realizó en el entorno de desarrollo integrado (IDE) de Arduino, específicamente en la versión 1.8.3 del mismo.

Dicho entorno posee las librerías y herramientas necesarias para la programación de todas las placas de la familia. Además cuenta con numerosos ejemplos de aplicación y librerías para los periféricos más usados y placas shield de ampliación.

La programación en este entorno se realiza en un lenguaje derivado de C orientado a objeto muy intuitivo y fácil de programar. Además de simple, la gran difusión de la plataforma hace que se encuentren en la web un sin número de ejemplos y aplicaciones.

Particularmente para nuestro proyecto se utilizaron las siguientes librerías:

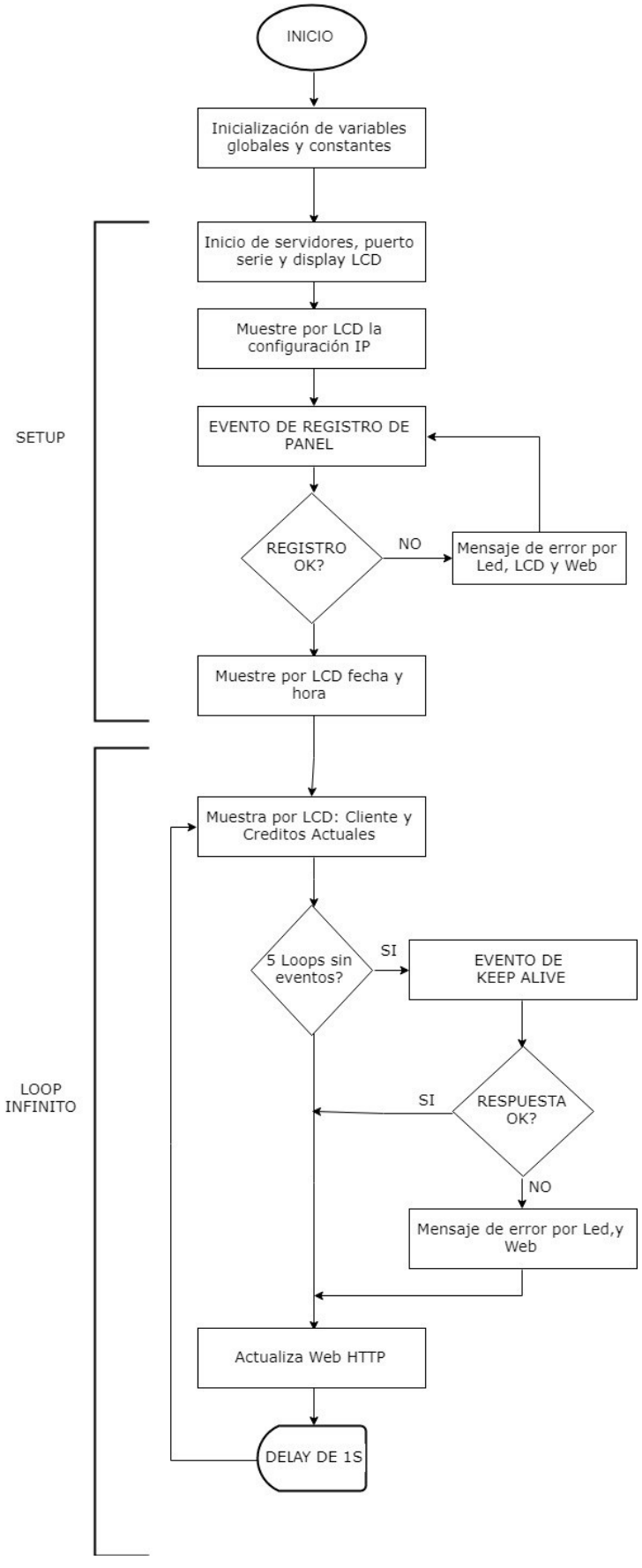
- **SPI.h**: realiza la inicialización y permite la utilización del puerto SPI del ATmega2560 a través del conector ICSP de 6 pines por el que se comunica el shield Ethernet.
- **Ethernet.h**: Posee la colección de comando de inicialización y operación para el shield basado en el W5100.
- **LiquidCrystal.h**: Esta librería posee los comandos de inicialización y operación para utilizar un display de cristal liquido basado en el controlador Hitachi HD44780 o compatible.

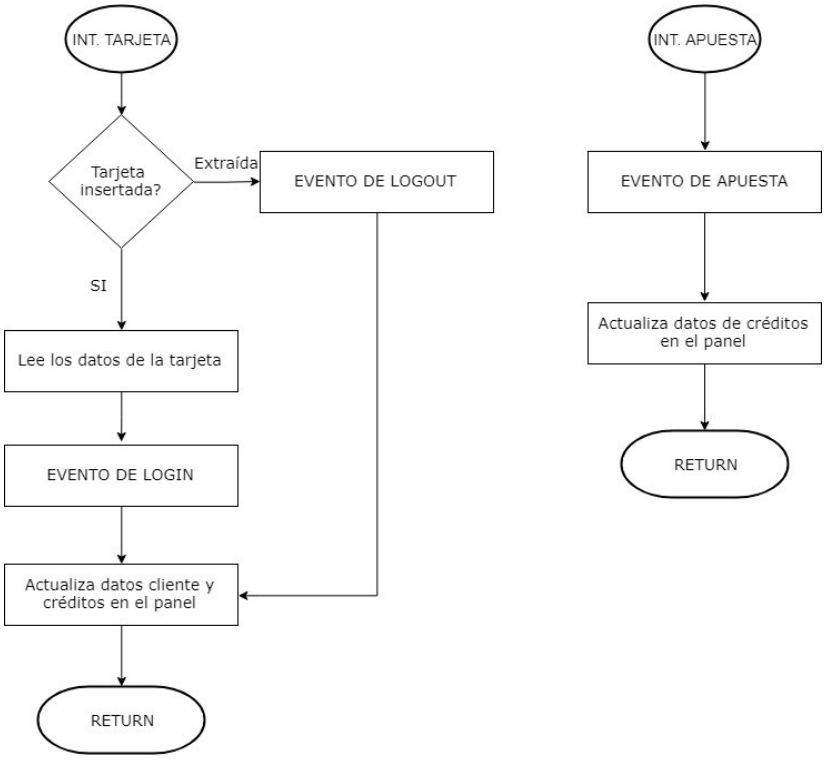
El software contempla como habíamos mencionado las siguientes funciones:

- Cliente TCP/IP
- Web server basado HTTP
- Comunicación serie para tarjetas
- Botones de apuesta simulando EGM

2.6.3 Diagrama de flujo del programa del PPT

Para explicar y entender el funcionamiento del programa realizado en el microcontrolador del panel, utilizamos el siguiente diagrama de flujo

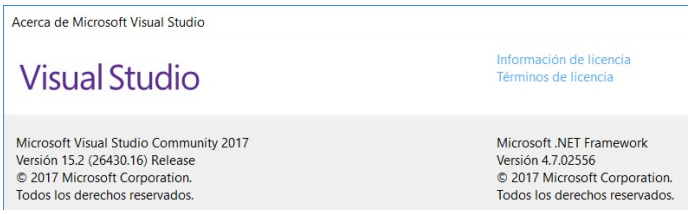

INICIO
Inicialización de variables globales y constantes
Inicio de servidores, puerto serie y display LCD
Muestre por LCD la configuración IP
EVENTO DE REGISTRO DE PANEL
REGISTRO OK?
NO
Mensaje de error por Led, LCD y Web
Muestre por LCD fecha y hora
SETUP
Muestra por LCD: Cliente y Creditos Actuales
5 Loops sin eventos?
SI
EVENTO DE KEEP ALIVE
RESPUESTA OK?
SI
NO
Mensaje de error por Led,y Web
Actualiza Web HTTP
DELAY DE 1S
LOOP INFINITO



El código completo implementado en el microcontrolador puede encontrarse en el anexo a este documento.

2.6 Desarrollo del Servidor de comunicaciones

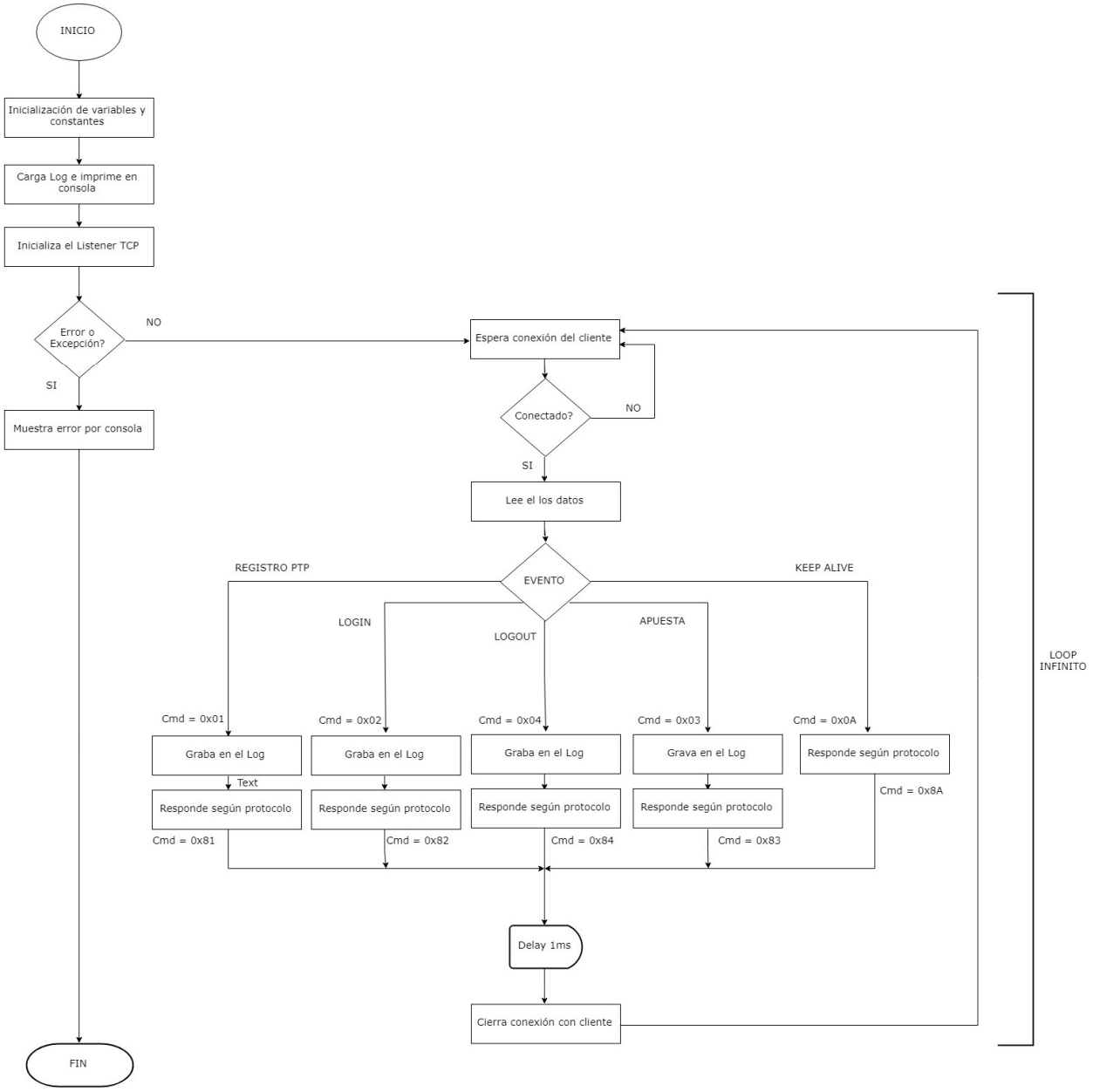
Para el desarrollo del servidor de comunicaciones, utilizamos la plataforma de desarrollo de Microsoft: Visual Studio. La versión Community 2017, que incluye .NET Framework 4.7 se ofrece de manera gratuita con limitaciones que no son un problema para nuestro desarrollo. En particular, utilizamos un lenguaje de programación orientado a objeto Visual C# 2017.



Para el desarrollo particular del servidor, utilizamos las librerías: System.Net y System.Net.Sockets que forman parte.NET Framework. En esta etapa del desarrollo, el servidor se comporta básicamente como un “Listener TCP” que escucha un puerto determinado y responde las peticiones que recibe por parte de los PPT. A su vez, realiza un log de cada evento recibido obviando los keep alive, en un archivo de texto plano.

2.6.1 Diagrama de flujo del programa del servidor

Para explicar el funcionamiento del programa realizado para el servidor, utilizamos el siguiente diagrama de flujo. El código completo está disponible en el anexo a este documento:



3. Prototipo y pruebas

En esta sección mostraremos las distintas instancias desarrolladas en funcionamiento con capturas de pantalla de las aplicaciones.

3.1 Prototipo del panel

El hardware del PPT en funcionamiento puede verse en la figura 3.1 donde se encuentran la placa MEGA2560 y el shield Ethernet montado sobre esta, el display LCD de 2x16 mostrando que no hay cliente ni créditos actuales y los botones para simular las apuestas. El cable amarillo corresponde al UTP de para la conexión Ethernet y el azul a la conexión USB para el simulador de tarjeta.

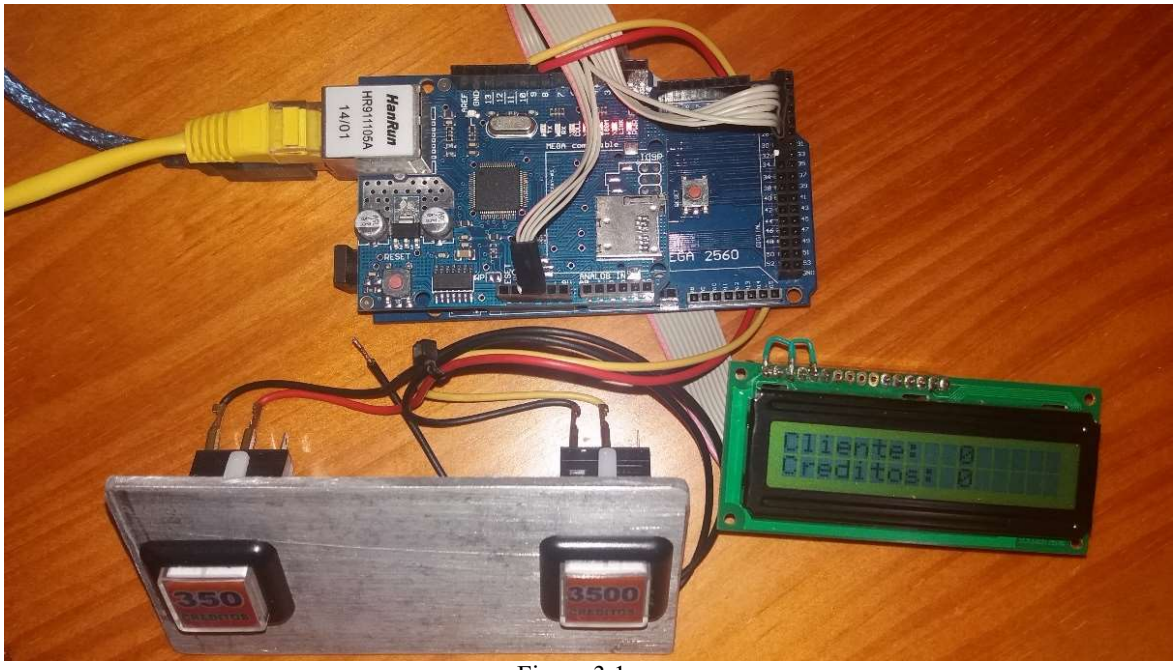


Figura 3.1

El simulador de tarjeta es una aplicación muy simple que permite emular la inserción y extracción de una tarjeta RFID a través del puerto USB de una pc. Por pantalla seleccionamos el evento que requerimos enviar al panel y en caso de la inserción se indica el número de tarjeta. Como función adicional permite consultar el esado del panel en cuanto si hay o no una tarjeta logueada. En la figura 3.2, vemos una captura de pantalla de la aplicación ejecutándose. El código completo de esta aplicación se encuentra disponible en los anexos de este trabajo.

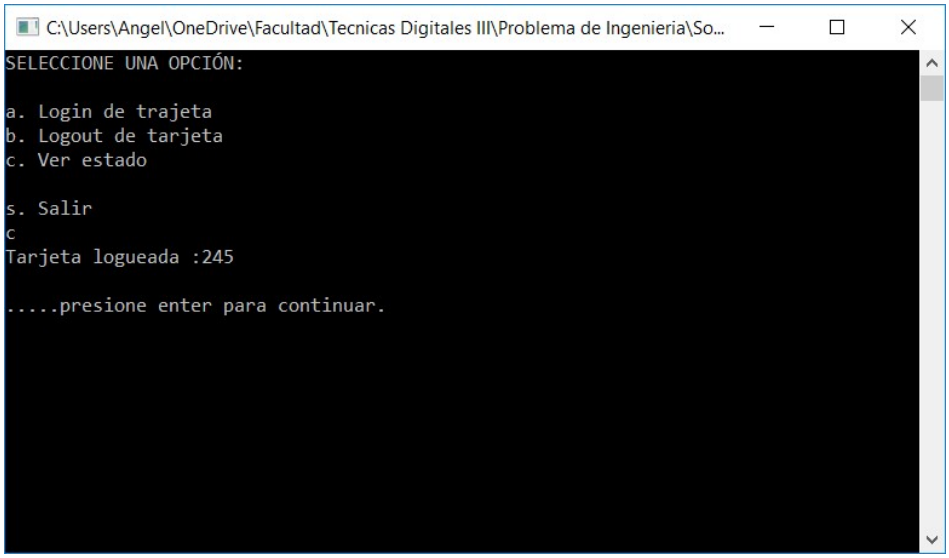


Figura 3.2

Otra de las funciones que mostramos en esta sección es la página web de mantenimiento o informe de estado que ofrece el PPT. La misma se accede mediante una pc conectada a la red. La figura 3.3 muestra una impresión de pantalla que corresponde a la página de mantenimiento provista por el PPT mediante el web server levantada desde un Chrome en una PC.

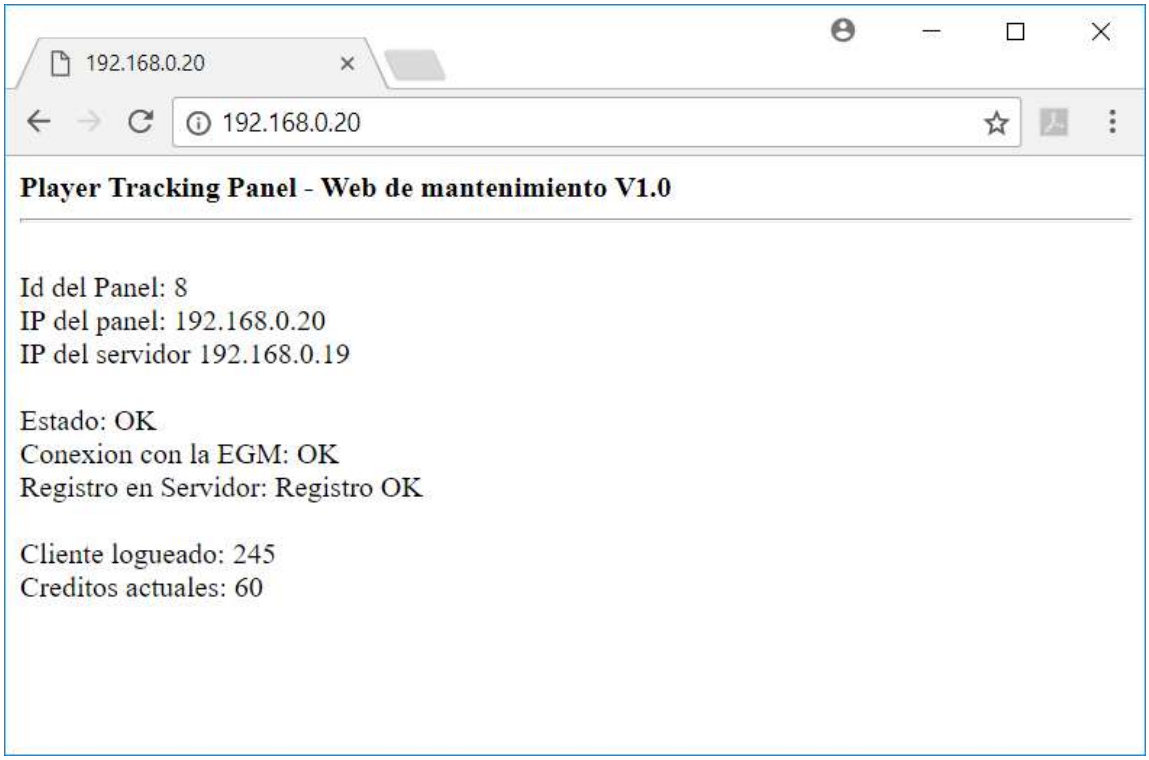


Figura 3.3

3.2 Prototipo de servidor

Se probó el servidor con un único panel conectado, en esta etapa de desarrollo es el hardware que hay disponible, la próxima instancia incluiría aumentar el número de clientes conectados. De todas maneras, para figurar el funcionamiento, capturamos la imagen de la consola donde el servidor muestra los eventos de comunicación que llegan y responde. En la figura 3.4 podemos ver una parta de esta comunicación donde forzamos los 6 eventos posibles con sus respuestas en hexadecimal.

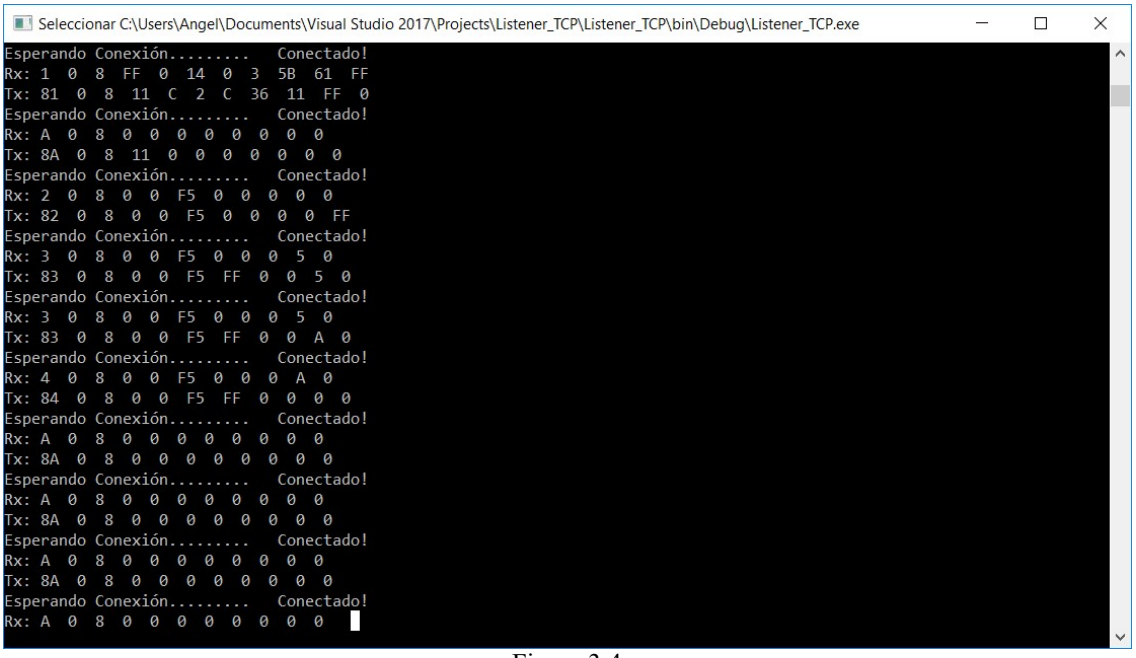


Figura 3.4

Contrastamos esto con el LOG creado en el archivo de texto plano en el servidor donde se decodifican los datos y se almacenan con referencia horaria. Así por ejemplo ante un evento de Apuesta (0x03 en hexadecimal) se almacena en el LOG la hora que se realizó, el número de cliente y la cantidad apostada. Esto puede verse en la figura 3.5.

Figura 3.5

4. Proyecto y costos

En esta sección analizamos los datos referidos a la implementación real del proyecto haciendo un pequeño análisis de costos que permitan establecer

4.1 Calculo de costo de implementación

En este punto, estimaremos el costo de implementación del sistema para 500 máquinas tragamonedas. Si bien los mismos serán estimados ya que algunos componentes serán necesarios especificar con más detalle, este valor nos permitirá tener una dimensión de los costos.

Componente	Costo Unitario	Cantidad	Total
Arduino mega2560 original Made in Italy	$ 1.200,00	500	$ 600.000,00
Shiel ethernet 5100 original Made in Italy	$ 1.500,00	500	$ 750.000,00
Display LCD 20 x 4	$ 300,00	500	$ 150.000,00
Gabinete Arduino mega	$ 250,00	500	$ 125.000,00
Fuente DC 12v 1A	$ 120,00	500	$ 60.000,00
Cable y conexiones	$ 100,00	500	$ 50.000,00
Panel de montaje en EGM	$ 750,00	500	$ 375.000,00
Hora de Mano de Obra de ensamble y montaje *	$ 354,00	2000	$ 708.000,00
PPT completo	**$ 5.636,00**	**500**	**$ 2.818.000,00**

* Se consideran 4hs para el ensamble y montaje de un panel por un técnico especializado a un valor hora de $354 (aprox. u$s20/h)

Componente	Costo Unitario	Cantidad	Total
Switch Hp 1410-24 24 Puertos 10/100 No Adm	$ 2.700,00	22	$ 59.400,00
Switch Hp Procurve 1410-24g J9561a 24-port Gigabit	$ 5.050,00	1	$ 5.050,00
Cable FTP cat5e multifilar marca AMP	$ 5.500,00	6	$ 33.000,00
Conectores RJ45 blindados AMP	$ 25,00	1200	$ 30.000,00
Hora Mano de Obra**	$ 354,00	200	$ 70.800,00
Infraestructura LAN			**$ 198.250,00**

** El cableado se realiza por conducciones ya montadas en la sala y entre muebles de las EGM preparados para esta tarea.

Componente	Costo Unitario	Cantidad	Total
Servidor HP Proliant DL380 Gen9 - E5-2640v4 – 16GB	$ 75.012,00	1	$ 75.012,00
HPE 600GB 12G SAS 10K 3.5in SC Enterprise HDD	$ 7.560,00	4	$ 30.240,00
HPE 500W Flex Slot Platinum Hot Plug Power Supply	$ 4.000,00	1	$ 4.000,00
HPE 3y 24x7 DL380 Gen9 FC SVC	$ 9.500,00	1	$ 9.500,00
WinSvrSTDCore 2016 SNGL OLP 16Lic NL CoreLic	$ 18.757,00	1	$ 18.757,00
SQLSvrStd 2016 SNGL OLP NL	$ 19.063,00	1	$ 19.063,00
Hora de instalación	$ 531,00	8	$ 4.248,00
Servidor			**$ 160.820,00**

*Tipo de cambio 1u$s = $17,7

El coso total de hardware, materiales y mano de obra para la implementación del sistema para 500 máquinas se estima en:

$3.177.000

4.2 Calculo de costo de desarrollo

En este trabajo se armaron los prototipos y los programas básicos para demostrar la capacidad funcional del sistema y establecer una base de diseño. Llevar el mismo a una etapa de aplicación requerirá sendas horas de desarrollo. Consultado con personas con experiencia en el desarrollo de este tipo de proyectos, pudimos realizar las siguientes estimaciones de costos de desarrollo.

Para la concreción del software completo del panel, incluyendo la programación del protocolo SAS para conectarnos con las EGM mediante el puerto serie, la programación del servidor de comunicaciones y base de datos más las aplicaciones de usuario final, se requerirían 2 programadores durante el plazo de 6 meses hasta tener una versión usable y 6 meses más para depuración de error y homologaciones.

Considerando un salario promedio de un programador con experiencia en $33.000 más cargas sociales, el costo total en RRHH seria de:

$1.110.000,00

4.3 Comparación con las soluciones disponibles en el mercado

Con el objeto de comprar y “ver donde estamos parados” para evaluar la viabilidad del proyecto, analizamos cuales serían las opciones de mercado con las que deberíamos competir:

Un sistema como el de Bally/SGS cotiza cada panel de player tracking y su interface de comunicaciones Mastercom en u$s1.100, los costos de servidores e infraestructuras no cambian mucho aunque generalmente requieren servidores más poderosos por ser sistemas más grandes y completo. Generalmente el sistema y el mantenimiento del mismo no se venden cobrando una cuota mensual que se establece por contrato y en muchas ocasiones siendo porcentajes de utilidades brutas. Esto obliga a la sala a sumar un socio mas al negocio en lugar de adquirir un bien. Solamente en interfaces PPT, para 500 máquinas estamos cerca de los $10 millones de pesos

Figura 4.1 Bally Mastercom

Una empresa nacional como Tecnoazar, opera de la misma manera, aunque el costo del panel es más económico, el mismo se encuentra cotizado en u$s600 llevando el total para 500maquinas a cera de $ 4,4 millones de pesos.

Podemos ver claramente que nuestro sistema se desarrollaría completo con menos dinero de lo que saldría comprar solo los paneles e interfaces de comunicación en otros proveedores, incluso nacionales. Este era el objetivo primario del problema.

5. Conclusiones

5.1 Sobre el proyecto

La conclusión de este trabajo es positiva ya que se consiguió la solución con los objetos primarios cumplidos. Si bien el proyecto es mucho más amplio de lo que se solucionó en este trabajo, pudimos cumplir con el alcance establecido. Consideramos que la solución presentada es una base aplicable a la solución final y que sirve como punto de partido para el proyecto en su totalidad.

A pesar del requerimiento multidisciplinario para llevar a cabo proyectos de esta envergadura, es función primordial del equipo de ingenieros a cargo del desarrollo conocer las instancias básicas y esenciales que hacen al funcionamiento del mismo, en este trabajo se investigaron y solucionaron cuestiones primarias del problema como la determinación del hardware y las especificaciones de red y equipamiento.

5.2 Valoración académica

Si bien el proyecto se planteó sobre una situación real y de aplicación real, el ámbito del mismo es académico, dentro de la materia Técnicas Digitales 3. Si bien toda la materia aporta conceptos que en mayor o menor medida hace posible la ejecución de este trabajo, nos permitimos destacar los principales temas aplicados de las unidades teóricas prácticas de la materia:

- CAPITULO 1 Los microprocesadores de 32/64 bits. Evolución. Lógica al azar vs. Microprogramación. Procesamiento paralelo. CISC vs RISC. Organización de memorias. Paginación y Segmentación. Memorias caché. Memoria virtual. Software de virtualización. Modos usuario y supervisor. Modo protegido. Multiprocesamiento. Coprocesadores.

- CAPITULO 3 Programación en lenguaje assembly en ambiente de PC. Sistema operativo DOS. Archivos. Directorios. Procesador de comandos. Funciones del DOS. Editor. Ensamblador. Directivas. Macros. Vinculador y localizador. Tabla de símbolos. Referencias cruzadas. Mapa. Herramientas de depuración de programas. Vinculación con lenguajes de alto nivel.

- CAPITULO 9 Redes de Computadoras. El modelo ISO-OSI. Topología de redes. Protocolos e Interfases. Placas de red. Hubs. Routers. Switches. Configuración.

Además, fueron necesarios aplicar conceptos vistos en materias anteriores de la carrera que consideramos fundamentales para el desarrollo y decidimos mencionar:

- Informática 1
- Informática 2
- Dispositivos electrónicos
- Tecnología electrónica
- Medio de Enlace
- Técnicas digitales 1
- Técnicas digitales 2

5.3 Referencias

Para la realización de este trabajo, se consultaron principalmente las siguientes fuentes:

✓ Apuntes de Técnicas Digitales 3 - http://www.frsn.utn.edu.ar/tecnicas3
✓ Foro de programación de Ardunio - https://forum.arduino.cc/
✓ Foros de programación de C# - https://social.msdn.microsoft.com
✓ Curso de programación on-line C# - https://codigofacilito.com/
✓ Manual de Protocolo de comunicación de EGM - http://slot-tech.ru/download/pdf/sas_602_protocol_manual.pdf
✓ Datos del instituto provincial de loterías y casinos - http://www.loteria.gba.gov.ar/

Fin del Informe.

Autor: Angel Famá 04-12-2017

6 ANEXO – Códigos de programas

6.1 Código del Servidor en C#

```
using System;
using System.IO;
using System.Net;
using System.Net.Sockets;
using System.Text;

namespace Listener_TCP
{
    class MyTcpListener
    {

        public static void Main()
        {
            clsClientes Cliente = new clsClientes();
            clsArchivos archivo = new clsArchivos();
            archivo.Cargar();
            TcpListener server = null;
            int creditos = 0;

            try
            {
                //inicialización del servidor
                Int32 port = 1300;
                IPAddress localAddr = IPAddress.Parse("192.168.0.19");
                server = new TcpListener(localAddr, port);
                server.Start();

                // Buffer para recibir y transmitir

                Byte[] RXdata = new Byte[11];
                Byte[] TXdata = new Byte[11];
                decimal Año, Mes, Dia, Hora, Minutos, Segundos;


                while (true)
                {

                    Console.Write("Esperando Conexión.........   ");
                    TcpClient client = server.AcceptTcpClient();
                    Console.WriteLine("Conectado!");
                    NetworkStream stream = client.GetStream();

                    //Lee los datos y los imprime en consola
                    int i = stream.Read(RXdata, 0, RXdata.Length);
                    Console.Write("Rx: ");
                    for (int k = 0; k < i; k++)
                    {
                        Console.Write("{0:X}  ", RXdata[k]);
                    }
                    Console.WriteLine(" ");

                    //Analisis e identificación de evento recivido, armado de respuesta

                    if (RXdata[0] == 0x01) //EVENTO DE REGISTRO DE PANEL
                    {
                        Cliente.IdCliente = RXdata[2];
                        Cliente.Creditos = 0;
                        Cliente.Evento = "Panel Registrado";
                        archivo.Guardar(Cliente);
                        TXdata[0] = 0x81;
                        TXdata[1] = RXdata[1];
                        TXdata[2] = RXdata[2];
                        Año = System.Convert.ToDecimal(DateTime.Now.ToString("yy"));
                        Mes = System.Convert.ToDecimal(DateTime.Now.ToString("MM"));
                        Dia = System.Convert.ToDecimal(DateTime.Now.ToString("dd"));
                        Hora = System.Convert.ToDecimal(DateTime.Now.ToString("hh"));
                        Minutos = System.Convert.ToDecimal(DateTime.Now.ToString("mm"));
```

```
                        Segundos = System.Convert.ToDecimal(DateTime.Now.ToString("ss"));
                        TXdata[3] = System.Convert.ToByte(Año);
                        TXdata[4] = System.Convert.ToByte(Mes);
                        TXdata[5] = System.Convert.ToByte(Dia);
                        TXdata[6] = System.Convert.ToByte(Hora);
                        TXdata[7] = System.Convert.ToByte(Minutos);
                        TXdata[8] = System.Convert.ToByte(Segundos);
                        TXdata[9] = 0xFF; //Status byte
                        TXdata[10] = 0x00;

                    }

                    if (RXdata[0] == 0x02) // EVENTO DE LOGIN DE CLIENTE
                    {
                        int ID = RXdata[5];
                        Cliente.IdCliente = ID;
                        Cliente.Creditos = 0;
                        Cliente.Evento = "Login";
                        archivo.Guardar(Cliente);
                        TXdata[0] = 0x82;
                        TXdata[1] = RXdata[1];
                        TXdata[2] = RXdata[2];
                        TXdata[3] = RXdata[3];
                        TXdata[4] = RXdata[4];
                        TXdata[5] = RXdata[5];
                        TXdata[6] = 0x00;//4 bytes para los crédito de del cliente
                        TXdata[7] = 0x00;//4 bytes para los crédito de del cliente
                        TXdata[8] = 0x00;//4 bytes para los crédito de del cliente
                        TXdata[9] = System.Convert.ToByte(creditos); //4 bytes para los crédito
de del cliente
                        TXdata[10] = 0xFF;//Status byte

                    }
                    if (RXdata[0] == 0x03) //EVENTO DE APUESTA REALIZADA
                    {
                        int ID = RXdata[5];
                        Cliente.IdCliente = ID;
                        Cliente.Creditos = RXdata[9];
                        Cliente.Evento = "Apuesta";
                        archivo.Guardar(Cliente);
                        creditos = creditos + RXdata[9];
                        TXdata[0] = 0x83;
                        TXdata[1] = RXdata[1];
                        TXdata[2] = RXdata[2];
                        TXdata[3] = RXdata[3];
                        TXdata[4] = RXdata[4];
                        TXdata[5] = RXdata[5];
                        TXdata[6] = 0xFF;//Status byte
                        TXdata[7] = 0x00;//NU
                        TXdata[8] = 0x00;//NU
                        TXdata[9] = System.Convert.ToByte(creditos);
                        TXdata[10] = 0x00;//NU


                    }

                    if (RXdata[0] == 0x04) //EVENTO DE LOGOUT DE CLIENTE
                    {
                        int ID = RXdata[5];
                        Cliente.IdCliente = ID;
                        Cliente.Creditos = RXdata[9];
                        Cliente.Evento = "Logout";
                        archivo.Guardar(Cliente);
                        creditos = 0;
                        TXdata[0] = 0x84;
                        TXdata[1] = RXdata[1];
                        TXdata[2] = RXdata[2];
                        TXdata[3] = RXdata[3];
                        TXdata[4] = RXdata[4];
                        TXdata[5] = RXdata[5];
                        TXdata[6] = 0xFF;//Status byte
                        TXdata[7] = 0x00;//NU
                        TXdata[8] = 0x00;//NU
```

```
                        TXdata[9] = 0x00;//NU
                        TXdata[10] = 0x00;//NU


                    }

                    if (RXdata[0] == 0x0A) //EVENTO DE KEEP ALIVE
                    {
                        TXdata[0] = 0x8A;
                        TXdata[1] = RXdata[1];
                        TXdata[2] = RXdata[2];
                        TXdata[6] = 0xFF;//Status byte
                        TXdata[4] = 0x00;//NU
                        TXdata[5] = 0x00;//NU
                        TXdata[6] = 0x00;//NU
                        TXdata[7] = 0x00;//NU
                        TXdata[8] = 0x00;//NU
                        TXdata[9] = 0x00;//NU
                        TXdata[10] = 0x00;//NU
                    }


                    // Envía la respuesta e imprime la misma en consola
                    stream.Write(TXdata, 0, TXdata.Length);
                    Console.Write("Tx: ");
                    for (int k = 0; k < TXdata.Length; k++)
                    {
                        Console.Write("{0:X}  ", TXdata[k]);
                    }
                    Console.WriteLine(" ");
                    System.Threading.Thread.Sleep(1);//Espera 1mS para terminar de transmitir
                    client.Close();
                }
            }
            catch (SocketException e)
            {
                Console.WriteLine("Excepción del Soket: {0}", e);
            }
            finally
            {
                server.Stop();
            }

            Console.WriteLine("\nServer terminado, presione enter para continuar.......");
            Console.Read();
        }
    }

   public class clsClientes
    {
        private int _IdCliente;
        private int _Creditos;
        private string _Evento;
        //encapsulamiento crtl+r+e
        public int IdCliente { get => _IdCliente; set => _IdCliente = value; }
        public int Creditos { get => _Creditos; set => _Creditos = value; }
        public string Evento { get => _Evento; set => _Evento = value; }
    }

    public class clsArchivos
     {
        public void Guardar(clsClientes item)
        {
            StreamWriter ar = new StreamWriter(@"C:\Users\Angel\Documents\000basePlayerT.log",
true);
            ar.WriteLine(DateTime.Now.ToString() + ";" + item.Evento + ";" + item.IdCliente +
";" + item.Creditos);
            ar.Close();
        }
        public void Cargar()
        {
            using (StreamReader sr = new
StreamReader(@"C:\Users\Angel\Documents\000basePlayerT.log"))
```

```
            {
                Console.WriteLine("****************** REGISTROS ACTUALES ******************");
                string line;
                while ((line = sr.ReadLine()) != null)
                {
                    Console.WriteLine(line);
                }
                Console.WriteLine("********************** ESO ES TODO **********************");
                Console.WriteLine("");
            }
        }
    }
}
```

6.2. Código del panel ARDUINO IDE

```
#include <SPI.h>
#include <Ethernet.h>
#include <LiquidCrystal.h>

#define LED LED_BUILTIN
#define BET1 2
#define BET2 3
#define LED LED_BUILTIN

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };//dirección MAC de la placa ethernet
byte ip[] = { 192, 168, 0, 20};                     //dirección IP del panel
byte subnet[]= { 255, 255, 0, 0};                   //Mascara de red
byte getway[] = { 192, 168, 0, 1};
byte Srv[] = { 192, 168, 0, 19 };                   //dirección IP del servidor
int port = 1300;
int HTTP_port = 80;
byte DataRX[11] = {};
byte DataTX[11] = {};
word PanelID=0x0008;
String CadenaSerie = "";
String SubCadenaSerie = "";
String Reg_date = "";
String Estado = "";
unsigned int ClientNow = 0;
unsigned int CreditNow = 0;
int TSC=0;
EthernetClient client;
EthernetServer server(HTTP_port);
LiquidCrystal lcd(34, 32, 30, 28, 26, 24);

void setup()
{
  pinMode(BET1, INPUT);
  pinMode(BET2, INPUT);
  attachInterrupt(digitalPinToInterrupt(BET1), bet, RISING);
  attachInterrupt(digitalPinToInterrupt(BET2), bet, RISING);

  pinMode(LED, OUTPUT);
  digitalWrite(LED, LOW);

  Ethernet.begin(mac, ip, getway, subnet);
  server.begin();
  delay(10);

  Serial.begin(9600);
  CadenaSerie.reserve(16);
  SubCadenaSerie.reserve(5);
  Reg_date.reserve(16);
  Estado.reserve(16);

  lcd.begin(16, 2);  // set up the LCD's number of columns and rows:
  lcd.print("Loc:");
  lcd.print(Ethernet.localIP());
  lcd.setCursor(0, 1);
```

```
  lcd.print("Srv:192.168.0.19");
  delay(2000);

  //*************** EVENTO DE REGISTRO DE PANEL ***************************
  // cmd 01| PTPid=01 status FF |IP 0 20 |ASET 220001 S.EGM FF|
  DataTX[0] = 0x01;
  DataTX[1] = highByte(PanelID);
  DataTX[2] = lowByte(PanelID);
  DataTX[3] = 0xFF;
  DataTX[4] = 0x00;
  DataTX[5] = 0x14;
  DataTX[6] = 0x00;
  DataTX[7] = 0x03;
  DataTX[8] = 0x5B;
  DataTX[9] = 0x61;
  DataTX[10] = 0xFF;

  do
  {
  lcd.clear();
  lcd.print("Id del Panel:");
  lcd.print(PanelID);
  lcd.setCursor(0, 1);
  lcd.print("REGISTRANDO SRVR");
  delay(1000);

  ServerCom(DataTX);

  if (DataRX[0]==0x81)
    {
    Reg_date="Registro OK";
    Estado="OK";
    lcd.clear();
    lcd.print("Registro ");
    lcd.setCursor(0, 1);
    lcd.print("Correcto");
    delay(1000);
    lcd.clear();
    lcd.print("FECHA: "); lcd.print(DataRX[3]); lcd.print("-"); lcd.print(DataRX[4]);
lcd.print("-"); lcd.print(DataRX[5]);
    lcd.setCursor(0, 1);
    lcd.print("HORA: "); lcd.print(DataRX[6]); lcd.print(":"); lcd.print(DataRX[7]);
lcd.print(":"); lcd.print(DataRX[8]);
    delay(2000);
    }
  else
    {
    Reg_date="Panel No Reg.";
    Estado="Sin conex. SRV";
    lcd.clear();
    lcd.print("Error Conx Srvr");
    lcd.setCursor(0, 1);
    lcd.print("NO registrado!!");
    WebServer();
    for (int i=1;i<5;i++){digitalWrite(LED, !digitalRead(LED));delay(300);}
    }
  }while (DataRX[0]!= 0x81);

 }

void loop()
  {
  TSC=TSC+1;
  lcd.clear();
  lcd.print("Cliente:  "); lcd.print(ClientNow);
  lcd.setCursor(0, 1);
  lcd.print("Creditos: "); lcd.print(CreditNow);
  WebServer();
  delay(1000);

  if (TSC>5){
      //**************EVENTO KEEP ALIVE******************
      // cmd 0A| PTPid=01 x 2 |
```

```
      DataTX[0] = 0x0A;
      DataTX[1] = highByte(PanelID);
      DataTX[2] = lowByte(PanelID);
      DataTX[3] = 0x00;
      DataTX[4] = 0x00;
      DataTX[5] = 0x00;
      DataTX[6] = 0x00;
      DataTX[7] = 0x00;
      DataTX[8] = 0x00;
      DataTX[9] = 0x00;
      DataTX[10] = 0x00;
      ServerCom(DataTX);
      if (DataRX[0]==0x8A){Estado="OK";digitalWrite(LED, LOW);}
      else {Estado="Sin conex. SRV";digitalWrite(LED, HIGH);}
      }
  }


//*************Comunicación con el servidor*****

void ServerCom( byte DataSend[])
{
  TSC=0;
  DataRX[0]=0x00;
  int i = 0;
  if (client.connect(Srv, port)) {
    //Serial.println("Conectado OK!");
    client.write(DataSend, 11);
    client.println();
  } else {
    //Serial.println("Fallo de Conexión con el Servidor");
    return;
  }
  while (client.connected())
  {
    if (client.available())
    {
      DataRX[i] = client.read();
      i++;
    }
    if (i == 11) break;
  }
  delay(5);
  client.stop();
  //Serial.println("Desconectado OK!");
}


//Evento serie para actividades de tarjeta

void serialEvent() {
  CadenaSerie = "";
  while (Serial.available())
  {
    // get the new byte:
    char inChar = (char)Serial.read();
    CadenaSerie += inChar;
  }
  switch (CadenaSerie[0])
  {
    case 'A':
      SubCadenaSerie = CadenaSerie.substring(2, 7);
      ClientNow = SubCadenaSerie.toInt();

      //**************EVENTO LOGIN CLIENTE******************
      DataTX[0] = 0x02;
      DataTX[1] = highByte(PanelID);
      DataTX[2] = lowByte(PanelID);
      DataTX[3] = 0x00;
      DataTX[4] = highByte(ClientNow);
      DataTX[5] = lowByte(ClientNow);
      DataTX[6] = 0x00;
      DataTX[7] = 0x00;
```

```
      DataTX[8] = 0x00;
      DataTX[9] = 0x00;
      DataTX[10] = 0x00;

      ServerCom(DataTX);
      CreditNow = word( DataRX[8],DataRX[9]);
      ClientNow = word( DataRX[4],DataRX[5]);


      break;
    case 'B':
      //**************EVENTO LOGOUT CLIENTE******************
      DataTX[0] = 0x04;
      DataTX[1] = highByte(PanelID);
      DataTX[2] = lowByte(PanelID);
      DataTX[3] = 0x00;
      DataTX[4] = highByte(ClientNow);
      DataTX[5] = lowByte(ClientNow);
      DataTX[6] = 0x00;
      DataTX[7] = 0x00;
      DataTX[8] = highByte(CreditNow);
      DataTX[9] = lowByte(CreditNow);
      DataTX[10] = 0x00;

      ServerCom(DataTX);
      if (DataRX[0]==0x84){
        CreditNow = 0;
        ClientNow = 0;
        }
    break;
    case 'C': Serial.println(ClientNow); break;
  }
}

//******************* Servidor WEB ************************
void WebServer() {

  // listen for incoming clients
  EthernetClient client = server.available();
  if (client) {
    boolean currentLineIsBlank = true;
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        if (c == '\n' && currentLineIsBlank) {
          // send a standard http response header
          client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/html");
          client.println("Connection: close");
          client.println();
          client.println("<!DOCTYPE HTML>");
          client.println("<html>");
          client.println("<strong> Player Tracking Panel - Web de mantenimiento V1.0</strong>");
          client.println( "<hr />");
          client.println( "<br>");
          client.println("Id del Panel:");
          client.println(PanelID);
          client.println( "<br>");
          client.println("IP del panel:");
          client.println(Ethernet.localIP());
          client.println( "<br>");
          client.println("IP del servidor");
          client.println("192.168.0.19");
          client.println( "<br>");
          client.println( "<br>");
          client.println("Estado:");
          client.println(Estado);
          client.println( "<br>");
          client.println("Conexion con la EGM:");
          client.println("OK");
          client.println( "<br>");
          client.println("Registro en Servidor:");
          client.println(Reg_date);
```

```
          client.println( "<br>");
          client.println( "<br>");
          client.println("Cliente logueado: ");
          client.println(ClientNow);
          client.println( "<br>");
          client.println("Creditos actuales:");
          client.println(CreditNow);
          client.println( "<br>");
          client.println("<br />");
          client.println("</html>");
          break;
        }
        if (c == '\n') {
          // you're starting a new line
          currentLineIsBlank = true;
        } else if (c != '\r') {
          // you've gotten a character on the current line
          currentLineIsBlank = false;
        }
      }
    }
    // give the web browser time to receive the data
    delay(1);
    // close the connection:
    client.stop();
  }
}

//************************ISR botones de apuesta*************
void bet(){
  delay(10);
  if (ClientNow!=0)EventoApuesta(5);
  }

void EventoApuesta(int apuesta)
  {

  //****************EVENTO DE APUESTA ****************************
  // cmd 03| PTPid=01 x 2 | cliente ID x 3 |apuesta x 4 |
      DataTX[0] = 0x03;
      DataTX[1] = highByte(PanelID);
      DataTX[2] = lowByte(PanelID);
      DataTX[3] = 0x00;
      DataTX[4] = highByte(ClientNow);
      DataTX[5] = lowByte(ClientNow);
      DataTX[6] = 0x00;
      DataTX[7] = 0x00;
      DataTX[8] = highByte(apuesta);
      DataTX[9] = lowByte(apuesta);
      DataTX[10] = 0x00;

    ServerCom(DataTX);

  if (DataRX[0]==0x83)
    {
      CreditNow = word( DataRX[8],DataRX[9]);
     }
  else
    {
    //almacena la apuesta para transmitir en el próximo evento
    }

  }
```

6.3. Código del simulador de tarjeta en C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO.Ports;

namespace SerialArduino
{
    class Program
    {
        static void Main(string[] args)
        {
            string opcion = string.Empty;
            string tarjeta = string.Empty;

            SerialPort miCom = new SerialPort();
            try
            {
                miCom.BaudRate = 9600;
                miCom.PortName = "COM4";
                miCom.Open();
            }
            catch (Exception)
            {
                Console.WriteLine("ERROR AL ABRIR EL PUERTO!!");
                opcion = "s";
                Console.WriteLine(".....presione enter para continuar.");
                Console.ReadLine();
            }
            while (opcion != "s")
            {
                Console.WriteLine("SELECCIONE UNA OPCIÓN:");
                Console.WriteLine();
                Console.WriteLine("a. Login de trajeta");
                Console.WriteLine("b. Logout de tarjeta");
                Console.WriteLine("c. Ver estado");
                Console.WriteLine();
                Console.WriteLine("s. Salir");
                opcion = Console.ReadLine();

                switch (opcion)
                {
                    case "a":
                        Console.WriteLine("");
                        Console.Write("NUMERO DE TARJETA:");
                        tarjeta = Console.ReadLine();
                        miCom.WriteLine("A");
                        miCom.WriteLine(tarjeta);
                        break;
                    case "b":
                        miCom.WriteLine("B");
                        break;
                    case "c":
                        miCom.WriteLine("C");
                        Console.WriteLine("Tarjeta logueada :" + miCom.ReadLine());
                        Console.WriteLine("");
                        Console.WriteLine(".....presione enter para continuar.");
                        Console.ReadLine();
                        break;
                    default:
                        break;
                }
                Console.Clear();
            }

        }
    }
}
```