AN INTERNET-BASED POINT-OF-BETTING (IPOB) FOR LOTTO PUNTERS

F.Nhenga-Mugarisanwa¹, L.Gunda² and F.T. Mangwanda³ A.M. Gwebu⁴
¹,²Department of Electrical and Electronic Engineering
Stellenbosch University, South Africa
¹fmugari@gmail.com
²17095808@sun.ac.za
³Department of Electronic Engineering
National University of Science and Technology, Bulawayo, Zimbabwe
mangwandatf@yahoo.com
⁴Department of Industrial and Manufacturing Engineering
National University of Science and Technology, Bulawayo, Zimbabwe
gwebuandilem@gmail.com

ABSTRACT

In this research an Ethernet-based lotto betting machine is developed, tested and deployed as an embedded web-client in a network. The device’s hardware mainly comprises a microcontroller, an Ethernet network interface shield, a keypad for input and an LCD screen output. The web-client sends requests to a remote Windows-Apache-MySQL-PHP (WAMP) server using PHP scripts. The Apache webserver serves the requests by querying the MySQL database where user identification, player-tokens and winning number records are stored and sends bet receipt SMS messages directly to users. To access the system a player enters a pre-purchased token number that allows them several chances via keypad. The remote server checks the token for authenticity, allows the user to enter a 6-digit number and registers the number in the database. It then sends a bet acknowledgement SMS while the LCD displays system transaction messages. This system eliminates the use of paper in betting slips, receipts and associated spoilt paper. It also replaces most human processes with inherent machine speed-up and continuous service availability. The system also reduces customer loitering and queuing, consequently eliminating crowding and associated environmental hazards. The system therefore potentially exhibits some positive sustainable environmental impact.
1 INTRODUCTION

The Internet-based Point-of-Betting (IPoB) system is an Ethernet embedded microcontroller lotto betting console that could eventually be implemented in Zimbabwe. The study’s targeted system is LOTTO™ and is the most popular in the country as it offers the highest jackpot payouts. LOTTO™ also features the majority of gaming outlets in the country with live draws held every Saturday on national television. To play the game a punter picks up a betting card with numbers 1 to 45 from a betting clerk’s counter typically located at the entrance of a major supermarket. The player then selects and crosses out any 6 of the 45 numbers, as shown in Figure 1, and submits the betting card to the clerk together with the payment.

Figure 1: Sample LOTTO™ Betting Card

The card is electronically scanned and the system automatically prints out the receipt showing the selected numbers. The system in its current form is laborious, time consuming and environmentally hazardous as spoilt betting cards are often thrown away by punters and, furthermore, some paper is rejected by the scanner system as it effectively works with flat surfaces. These factors are also unfavourable for business growth as new clients are discouraged by long queues and poor service time. The number of customers also directly impacts on community development in the long term. This is mainly because the Lotteries Gaming Act Chapter 10 (clause 45) of Zimbabwe [1] prescribes that the lotto business, must contribute some 25% of its revenue toward social service, public welfare or relief of distress within Zimbabwe.

This research seeks to improve the whole system performance and offer a sound return on investment over a reasonably short period of time. This will be made possible by introducing an interactive betting console that connects to a database over an Ethernet network.

This paper briefly examines similar systems in form of patent applications and states the main aim of the study and system objectives. This is followed by the methodology section which describes system hardware and software development that then leads to the system integration and testing section, culminating in deployment and concludes with some sample results. The research is of a practical nature and hopes the system may eventually be implemented or some of the ideas adopted in similar or different systems.

2 SOME RELATED RESEARCH

The authors examined some of the research and developmental work in the field of lottery technology to determine the trends and possibly identify any gaps in them that they could fill. Work by Piper and Smith [2]; drew many technological similarities with that of De Cuba
[3], Choi and Park [4] and several other researchers. The following section briefly describes the work by these three teams of researchers.

2.1 Lottery System

The project by Piper and Smith [2], aptly named Lottery System, describes a computer based system which allows entries to be sold over the telephone, or by using other systems such as an ATM or POS machine, an email, and, via kiosks. The Lottery System participants are invited to choose at least one unique number from a defined range of numbers, for example a number between one and one million. The participant can register their selection with an entry-logging engine. This engine records the details of the participant (for example telephone number or email address), the number selected, the date and time of the entry, and, an optional receipt number. The engine then passes each entry in turn to a lottery engine, which allows the competition to run until a winner is determined.

2.2 SMS Messaging System Accommodation Variable Entries for Lotteries System

In this system by De Cuba [3], an SMS lottery gateway server obtains one SMS message from a mobile phone and parses it. There are various ways in which the SMS message can be parsed such as looking for a number (which indicates the number of entries) and an associated pick. Lottery winnings can be credited to a prepaid or post-paid account.

2.3 Lottery Business System and the Working Method Using the Personal Unit on the Wire/Wireless Network

In their patent documentation, Choi and Park [4] propose the use of an existent IT infrastructure to run a lottery business system. The two indicate that use of the IT backbone secures both domestic and international competitiveness, resulting in improved investment efficiency. Their system uses a personal terminal or a versatile TV terminal (which has a two-way function) to access a public switched telecommunication network or a broadcasting network. The punters have to access the network to purchase a lottery ticket and check the winning ticket.

3 AIM AND OBJECTIVES

3.1 Aim

The aim was to design a functionally lightweight embedded web client system that could be used as an automated standalone lotto booth, allowing users to submit their bets over the Internet and receive bet acknowledgements without filling in any paperwork and with minimum or no human assistance.

3.2 Objectives

The objectives were to deliver a system that easily fits into the existing system network with the following major features:

- The system should be easy to operate and maintain such that it can be easily installed.
- The system should be user friendly and fully interactive guiding the user step-by-step as they log in data in order to reduce operational error and enhance its life cycle.
- The system should send data in the correct Hyper Text Transfer Protocol HTTP/1.1 format, conform to IEEE802.3 Ethernet standard and to Internet Engineering Task Force (IETF) technical recommendations, Fielding et al [5]
- The system should be robust such that it can smoothly handle errors and exceptions.
The system should be loosely coupled from the main LOTTO™ system so that local maintenance does not affect the main system.

The system should be extensible so that any changes to the LOTTO business rules will not shut the system down but involve minor or no software adjustments.

4 SYSTEM ARCHITECTURE

The system development strategy is aimed at replacing the current manual system without disrupting normal business processes, thus the new system’s functionality should map to the existing system. The keypad represents the LOTTO™ betting card and the number selection is done by keying in the keypad number keys. The customer and betting clerk’s interactions are replaced by a Liquid Crystal Display LCD display of, among others, a welcome screen; the echo of the pressed keys; step-by-step user instructions and error reporting; all these displays providing real-time feedback driven by events within the local system or from the remote server responses.

![Figure 2: Overall LOTTO™ System Deployment Diagram.](image)

The deployment diagram in Figure 2 illustrates the IPoB system’s location within the LOTTO™ system. The current system serves many different web clients from different devices including mobile phones, using web browsers like Google Chrome among others. A bet placed by a client is a request to the server for some file, which the server pushes back to the client through his or her browser for viewing. The IPoB will send a keypad input formatted as a client request to the server and when the response arrives back the system will be configured to interpret and display the information on the console screen. The IPoB will thus emulate a low level web browser through which punters can place their bets and get SMSes from the remote system as shown in Figure 2. The remote engine still remains the lotto draw centre access the system as any other player.

4.1 IPoB System Hardware Components Description

The IPoB system’s architecture is represented by the stacked hardware model shown on the left hand side in Figure 3. The magnified illustration of the WIZnet® W5100 Ethernet controller block diagram is shown on the right with numbers indicating the applicable hardware component drawn from the hardware side to show where each hardware application approximately fits into the W5100 block stack diagram, extracted from the datasheet [6].
4.1.1 Microcontroller

The microcontroller used the Atmel AVR® ATmega328 which is housed on an Arduino™ board prototyping platform. According to Banzi [7], the Arduino hardware and software are available as open source under the Creative Commons (CC) licensing agreement that allows for free circuit reproduction. Under this licensing agreement individuals can make Arduino-compatible boards, which can be exact copies of the original, free of charge using freely downloadable printed circuit board templates.

Creative Commons has been described by Broussard [8] as being at the forefront of the “copyleft” movement, which seeks to support the building of a richer public domain by providing an alternative to the automatic “all rights reserved” copyright, and has been dubbed “some rights reserved”.

The IPoB system used the Arduino Uno Revision 3 (or R3) shown in Figure 4. The Arduino Uno Revision 3 (or R3) board features an ATmega328 microcontroller on board, like other Arduinos, but differs from all preceding boards in that its USB-to-Serial converter chip uses the more powerful ATmega16U2 instead of the 8U2, has a highly sensitive RESET circuit and input output reference (IOREF) that allows external shields to adapt to the voltage provided from the board automatically.

The top two headers have fourteen digital input/output (IO) pins marked on the board as pins 0 to 13, six of these digital pins numbered 3, 5, 6, 9, 10, and 11, and marked “~” can, through software configuration, also be turned into six analogue output pins using pulse-width modulation (PWM) in which rapid pulses can be output to control motors at various

Figure 3: System Hardware Stack WIZnet W5100 Protocol Stack [6]
speeds, or play music (by varying the pulse frequency). The right bottom header has six Analogue input pins (marked as pins A0 to A5) which are dedicated analogue input pins that take analogue values (such as voltage readings from a sensor) and convert them into a number between 0 and 1023. The rest of the pins are used for powering external devices during development, or other Arduino accessory boards called shields which have pins that have the same form-factor as the basic Arduino development board. Figure 6 shows an Ethernet shield connected on top of the Arduino via the pins-to-headers coupling. The In Circuit Serial Programming (ICSP) pin terminal block on the right edge of the board provides the Serial Peripheral Interface (SPI) bus interface for connecting with the Ethernet Controller and, can also be used for programming the Arduino from an external programmer.

![Arduino Uno R3 System](image)

**Figure 4: Arduino™ Uno R3 System [9]**

The board can be powered from the host computer’s USB port, or from most USB chargers, or an AC adapter (9 volts is recommended). Banzi [7] describes how the power to the Arduino can be connected via USB. He concedes that in the absence of a normal power supply plugged into the board’s power socket, power will be derived from the USB input and the board automatically changes over to the mains input when it becomes available.

### 4.1.2 The Keypad

The IPoB system’s input device is a 4x3 keypad with a construction similar to the ones used on telephone systems whose digit ‘5’ key is marked to assist the blind and visually impaired people to locate the numeric keys by the sense of touch. The keypad is a matrix of switches that connect at the intersection of row–column conductor lines as shown in Figure 5. To use the keypad the rows and column pins must be identified using the manufacturer’s datasheet or manually by depressing each key and testing for circuit continuity between the pins using an ohmmeter. It must be noted that the keypad layouts in Figure 5 only serves as an illustration others may differ from it.
4.1.3 The LCD Display

The LCD used in the project is a 20 x 4 alphanumeric display. An example by Margolis [10] was used as reference to determine the LCD pin assignments which are VIN and Vout (used for the contrast), Read/Write (R/W), Reset (RS) and enable (E) and the data pins D4 to D7. The data connection uses the 4-bit mode (which excludes data pins D0 to D3) instead of the 8-bit in order to use fewer pins on the Arduino. The display module is used in conjunction with the keypad above to echo each key value as it was entered. The LCD also echoes instructions, transaction acknowledgements and errors generated by the remote server.

4.1.4 The Serial Peripheral Interface

The microcontroller links with the Ethernet Controller via the SPI bus. SPI is a four-wire synchronous serial bus protocol developed by Motorola and integrated in many of their microcontrollers. SPI bus consists of four signals: master out slave in (MOSI), master in slave out (MISO), serial clock (SCK), and active low slave (or chip) select (SS). As a multi-master/slave protocol, communications between the master and selected slave use the unidirectional MISO and MOSI lines, to achieve data rates in full duplex mode. With SPI the maximum number of devices that the master can connect to is limited by main microcontroller’s available pins. Sandya and Rajasekhar [11], in their paper on the design and verification of SPI, also demonstrate that using SPI ensures that the speed of data transfer is much faster.

4.1.5 Ethernet Shield and Magnetics

The WIZnet™ W5100, whose internal structure is shown in Figure 3, is the main component of the Arduino Ethernet Shield. The W5100 implements the Transport Control Protocol/Internet Protocol (TCP/IP) protocol suite in hardware for fast response. The Arduino Ethernet shield connects to the Arduino Uno using an SPI interface via the In Circuit Serial Programmer (ICSP) header as shown in Figure 4. The shield module and the Uno have the same number of pins and headers thus it fits directly on top of the Uno as shown in Figure 6. The Magnetics part is made up of a transformer, an RJ-45 socket and link establishment LEDs thus the meshing of the Arduino to Ethernet shield completes the hardware stack in Figure 3.

4.2 System Software Architecture and Components

The software architecture closely follows the protocol stack model shown in Figure 3 illustrated by the W5100 block diagram. The application at the top can be viewed as the Arduino microcontroller system’s data, for example bet number entries, which it
encapsulates within an HTTP GET request as a socket (that is a host destination address/ port number pair).

The driver section is accomplished by the Arduino-SPI messages to W5100 that enable flow of data under the control of the Arduino. In the W5100 module the data are further encapsulated within TCP/IP, MAC using the data from the Arduino. Other networking information is added at the physical or PHY layer level where physical receive/transmit negotiation takes place as the data are sent as electrical pulses in packets according to IEEE802.3 standards.

The software development is carried out using the Arduino Integrated Development Environment (IDE) to write program code, compile, debug and run all under the same environment. The software for programming the Arduino is, according to Igoe [12], easy to use and also freely available for Windows, Mac, and Linux computers, at no cost.

4.2.1 The Arduino Sketch

The basic Arduino program is called a sketch (shown in Figure 7) and is written using a programming language called Wiring. The Arduino Integrated Development Environment (IDE) is used to create the sketches. A sketch is written and compiled (or verified in Arduino language) to check for any syntax errors and uploaded onto the Arduino board to run. One very useful IDE tool is the Serial monitor which print out the Arduino board’s output on the screen, this can be used for debugging by example embedding some printing routine at some point within a loop to verify that it executes.

The anatomy of a sketch is in three parts namely the declaration of variables and constants; then two functions namely setup () and loop() function. During program execution the setup() runs once during start up or upon reset then the loop() follows and runs for as long as the Arduino is powered on.

Wiring is a version of the C++ language which the Arduino developers have simplified by the inclusion of libraries called Hardware Abstraction Libraries (HALs). The HAL is basically a class that has been prewritten with the generic features for Arduino hardware accessories to ease the process of interfacing the hardware to the Arduino platform.
HALs provide software code wrappers that hide the low level software implementation details of the hardware turning the implementation into that of any ordinary C++ class. The HAL is usually implemented in a sketch as a header file (with an .h extension) by importing it into the sketch. In this project four libraries were used and these are briefly described below.

4.2.2 The Keypad.h Library

The Keypad.h library contains the keypad’s generic key layout description in terms of rows and columns. The user must consult the manufacturer’s information about the key-to-pin mapping or test it with an ohmmeter. The developer programmatically maps the keypad pins to the Arduino digital pins as inputs. Once configured and declared the keypad object can then be used like any other C++ class to trigger events such as when a key is pressed.

4.2.3 The LiquidCrystal.h Library

The LiquidCrystal.h library is used with the LCD which connects to the Arduino digital pins in the output mode. Its configuration includes the number of characters columns and rows; the bit mode, that is the number of data pins whether 4-bit or 8-bit. The implementation includes the cursor starting position, blinking, outputting a message using an lcd.print (“a message”) function.

4.2.4 The SPI.h Library

The SPI.h library is used to control data between the Arduino (as the master) and any other devices (as the slaves) using chip select and MISO/MOSI routines. In this project it is used to transfer HTTP-bound data from the Arduino (master) to the W5100 Ethernet controller which is the slave.
4.2.5 The Ethernet.h Library

The Ethernet.h library contains both several network-related classes including the EthernetClient and EthernetServer this paper is confined to the EthernetClient class. The EthernetClient instance is configured to transfer TCP/IP sockets to a server by assigning a mac address which comes with the hardware, setting up the server IP address, and, if a router is used, the gateway and subnet addresses. Once configured the EthernetClient object can use the EthernetClient.connect (server, portnumber) method to connect to the server. If the connection is established the EthernetClient object can then just use a println method to send HTTP requests using a GET method such as:

```
EthernetClient.println("GET /search?q=arduino HTTP/1.1");
EthernetClient.println("Host: www.google.com");
```

5 IMPLEMENTATION - PUTTING IT ALL TOGETHER

5.1 The Keypad Implementation and Testing

The keypad’s pins Figure 5 were checked to identify which ones of them were rows and columns using a continuity tester. After identification the four row pins were connected to the Arduino digital pins 8,7,6,5 and the three columns to pins 4, 3 and 2. The Keypad.h library described in 4.3.1 above was imported to the IDE Arduino sketch and the IDE’s Serial.println() used to output any key press on the monitor window. An excerpt of the code (with comments after the “//”) is as follows:

```
//The Keypad class constructor initialises a keypad object to create a virtual keypad
//that maps the physical hardware rows (ROWS) and columns (COLS)
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS);
char key = keypad.getKey(); // get the key face value
if (key != NO_KEY){
    Serial.println(key); // show the face value key that was pressed
}
```

5.2 The LCD Implementation and Testing

The LCD display connections were obtained from the examples by Margolis [10] and the program was implemented using the LiquidCrystal.h library to create a “4x20”character lcd object using the 4-bit mode to save pins as described in 4.3.2 above. The following code snippet displays the message “To PLAY!!!” on the forth row starting from the sixth column:

```
// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(A0, A1, A2, A3, A4);//These are the Arduino Analogue Out pins
// which will connect with LCD for RS, E, D7, D6, D5, D4 respectively
void setup() { // set up the LCD's number of columns and rows:  
lcd.begin(20, 4);// 20 Columns or characters, 4 Rows }  
void loop() {lcd.setCursor(5, 3);//cursor set to the sixth column and fourth row 
lcd.print(" TO PLAY!!!");
delay(500); }
```
Figure 8: The IPoB Console

Figure 8 shows the implementation of the above program code displayed on the fourth row.

5.3 The Input Console Integration and Testing

One of the objectives outlined in subsection 3.2 that the system must be fully interactive meant that the keypad inputs must be echoed on the LCD display. This was solved by combining the LCD and keypad sketches, then modifying the lcd object to print the value of the pressed key using the following sample code:

```cpp
char key = keypad.getKey(); // get the key face value
if (key != NO_KEY){
    lcd.println(key); // show the face value key that was pressed on LCD display
}
```

5.4 Ethernet Implementation and Testing

The Ethernet shield comes with a Media Access Control (MAC) address which is a unique hardware address for initial identification on the network. This address is then mapped to the more universal IP address, the datasheet WIZnet [6] provides more detail. To set up a network connection the Ethernet.h library imported to a sketch with.

Connecting to a server whose IP address is already assigned requires the:

```cpp
client.connect(serverAddress, 80); // this is a TCP socket to an HTTP port address
```

The message that is sends a user’s mobile phone number could be coded as an HTTP request of the general form:

```cpp
client.println("GET/mobileNumber.php?mobile_number="0773122110" HTTP/1.1");
client.println(serverAddress); // remote server name
```

The response from the server is read then printed to the LCD in the form:
If (client.available()){
    char in_msg = client.read();
    lcd.print(in_msg);
}

5.5 IPoB HTTP GET Request and Server Side Implementation

In this project the embedded web client sends keypad input data as HTTP GET requests to the remote server to be processed by a targeted PHP file. The following example shows the implementation of the token number for validation request from a client object called client to a remote server called serverAddress:

```java
client.print("GET/ serverAddress/validat_token_number.php?
    token_number="number");
client.print("HTTP/1.1");
```

The above request sends the parameter token_number and its value “number” to the remote WAMP server called serverAddress and then passed on to the validat_token_number.php file where it is assigned to the PHP $_GET variable. The PHP file processes the value by executing a database query checking through the MySQL database for token number records. The result is returned with a PHP echo of “Y” or “N” or “S” (as listed in Table 1), embedded in HTML.

5.5.1 Server Response Implementation

The server’s response to an HTTP request consists of a header with some code for example HTTP/1.1 200 OK for a successful request or HTTP/1.1 404 Not Found for an unsuccessful web page request. The header also contains many other server description parameters and an HTML body, as given in the reference documentation by Fielding et al [5]. In this research the system analysed the content of a successful request otherwise displayed an error and a retry message giving the user two more chances. (Table 1 provides a list of messages).
Figure 9: A Sample Flow Chart

6 DEPLOYMENT
The prototype was assembled and connected via Ethernet to a laptop that had a WAMP server with PHP scripts prepared by a colleague and a GSM modem. A five-field MySQL database table was created with a filled in token number field and blank fields for mobile phone, bets, DateTime and transactionID numbers. To test the system users were given a piece of paper with a copy of the token number and asked to place a bet, the system responded well and upon success users received an acknowledgement SMS on their mobile phones. The system test scenario was further extended to scores of high school students during the university open Day and proved quite popular with a virtual queue forming up as more students learnt about it. The downside was that some of the acknowledgements SMSes were delivered days after the bets.

7 RESULTS
In order to verify that the system was running according to specification tests were undertaken. Hardware tests were carried out to ensure that the system delivered the appropriate responses from the server as shown on the table below. Erroneous entries were also made to the system to carry out limited stress testing.
# Table 1: Remote System Response Codes

<table>
<thead>
<tr>
<th>Server Response Message</th>
<th>HTML Message</th>
<th>Meaning</th>
<th>IPoB LCD Message Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Y”</td>
<td></td>
<td>12 digit scratch or token number valid</td>
<td>“Enter Mobile Phone Number”</td>
</tr>
<tr>
<td>“N”</td>
<td></td>
<td>token number invalid (User is given 3 chances)</td>
<td>“Sorry Token Not Found”</td>
</tr>
<tr>
<td>“C”</td>
<td></td>
<td>Database record update failure because token already used. (User is given 3 chances)</td>
<td>“Sorry Token Used”</td>
</tr>
<tr>
<td>“A”</td>
<td></td>
<td>Mobile number not valid for network.</td>
<td>“Mobile Number out of Range”</td>
</tr>
<tr>
<td>“M”</td>
<td></td>
<td>Lotto first number entered is out of the 1 to 45 range.</td>
<td>“First bet out of 1 to 45 range”</td>
</tr>
<tr>
<td>“H”,”D”, “E”, “F”, “G”</td>
<td></td>
<td>As above for lotto second to sixth number entries respectively out of the valid range.</td>
<td>“Second [third...up to sixth] bet out of 1 to 45”</td>
</tr>
<tr>
<td>“J”</td>
<td></td>
<td>Repeated number entries detected</td>
<td>“Repeated Bet entries please check”</td>
</tr>
<tr>
<td>“S”</td>
<td></td>
<td>Successful transaction.</td>
<td>“Bet successful SMS to be sent shortly”</td>
</tr>
</tbody>
</table>

## 8 CONCLUSION

The system console shown in Figure 8 was successfully tested by scores of “clients” comprising groups of high school students and teachers during university open Day. From that exposure the authors draw the conclusions that:

- The system proved easy to set up, operate and is user-friendly. Users got a fully interactive step-by-step guidance as they logged in data and the main problems encountered such as undelivered SMSes or network disruptions were not attributable to the system.
- The system was constructed using Arduino components which conform to IEEE802.3 Ethernet standard. It was also able to send data in the correct Hyper Text Transfer Protocol HTTP/1.1 format as per Internet Engineering Task Force (IETF) technical recommendations, Fielding et al [5] implementation
- The system was robust enough as it could inform users of errors and what they should do as shown by the Table 1 and the flow chart in Figure 9.
The system design was such that it behaved like any web client that sends standard HTTP requests thus it can be used as a browser that submits HTML web form data thus it can be easily be ported to other systems and is loosely coupled from the main LOTTO™ system so that local maintenance does not affect the main system.

9 REFERENCES


