

System Block Documentation

Principle of Operation:

A block diagram for the RFID Marking management solution is shown in Figure 1. We have separated the hardware components from the software components. The hardware components are on the left and the software components are on the right.

In this design our intent is to make parking easier for the busy person in congested city areas. To provide a viable solution we must interface the hardware with the software flawlessly. Once the PC receives information about each car in the parking space, the rest of the processing of data is done in software.

Hardware:

The system starts functioning when the customer parks in a parking space with an RFID reader. The RFID reader is constantly emanating its signal. This electromagnetic field is able to power the passive RFID tag when a car enters the parking spot. The motorist need not do anything more for communication to take place.

The RFID tag then communicates its unique 64bit ID to the RFID reader and the time the car enters the spot is transferred to the PC via the PIC and recorded in the MySQL database. When the car leaves the spot, the tag is deactivated and the reader will not be able to detect the tag, and determine that a car is no longer parked in the spot. The RFID reader sends information that the car has left to the PC via the PIC. The time of the departure of the car is stored in the database. At a later date we can now compute how long the car was parked.

Software:

The software component of this design includes a database, web server and an Simple Mail Transer Protocol (SMTP) email server. The purpose of the software is to provide the user a point and click interface to allow them to view information on the current state of their parking spot or to see where an open parking spot is available. On the website they can also check their account status and find out how much time that they have been parked for.

Specification of Blocks:

There are seven blocks in the block diagram of the RFID Parking Management solution. The boxes in the left half of the diagram indicate each component in the hardware interface. The arrows indicate the communication between the other hardware components in the diagram.

The boxes in right half of the diagram indicate each component in the software interface. The arrows indicate the data flow between each software component. The ovals indicate examples of the queries that will be made from the website.

RFID Tag Block:

The RFID tag block has one function: To identify to the RFID reader that a car is parked in its spot. The tag will be placed on the front windshield where it will be within 1 meter of the parking spot and will operate at 13.56MHz, matched with the reader. The RFID tag is passive, meaning that when it comes in proximity of the RFID reader, it is powered by the electromagnetic field of the reader. The tag then communicates its unique 64-bit identification number.

RFID Reader Block:

The RFID reader has an antenna which governs the distance it can read an RFID tag. The read distances are 1 meter, with a 5V, 200mW signal at 13.56MHz. The RFID tag is then identified by the reader.

After the RFID tag communicates its unique 64-bit identification number the reader informs the PC that a car has entered the spot. The time of entry is stored to the PC's MySQL database. Every 60 seconds the RFID reader will attempt to communicate with the tag to see if it is still there. If the reader can still communicate with the tag, the reader will take no further action. If the reader does not find the tag and communication times out, the reader sends a signal to the PC informing it that the car has left the spot.

PIC Block:

The PIC block controls the RFID reader IC through an SPI interface, which is a 3 wire serial interface. The three wires include: SCLOCK, D_IN and D_OUT. SCLOCK is a bi-directional pin which is shared between the PIC controller and the RFID Transceiver IC. D_IN is the data into the PIC, and D_OUT is data out to the RFID reader IC. The SPI interface's specifications are well documented in the TI Reference guide and the PIC16F877A manual. For the PIC to gain control the RFID reader it must pulse D_IN.

After the PIC receives the data on D_OUT, it is sent to the MAX232 Serial Line driver which converts TTL voltages of 0V/5V to RS232 voltages of -12V/12V. The output of the MAX232 chip is connected to the PC.

Currently our solution uses an RS232 interface. We would like to use RS485 and if time permits we will experiment with that.

PC Serial Communications Block:

The PC block is a large part of the system as it is running all the software components of the solution. The system block design for the PC software is shown on the right half of the diagram. The PC's speed in our solution needs to be at least 500MHz, with 256Mb RAM, and a 40gb hard drive space for the database storage. The PC will be outfitted with a flavor of Linux, which will most likely be Gentoo.

The PC receives data from the PIC using an RS232 interface, which has been converted from TTL voltages to RS232 voltages. The PIC and PC communicate using two pins, one for transmission (TX) and one for receiving (RX). Most of the time the PIC will be sending data to the PC, the PC really doesn't have any need to send data to the PIC.

Once the data is on the PC, it can then be merged into the MySQL database. The data can immediately be queried once it enters the database.

We would like to use the RS485 interface if time permits:

This interface allows for up to 32 different readers to be connected to the same PC. This simplifies management of the parking meters as they can all be hooked up to the same interface and their data collected by one PC. The RS485 interface uses twisted pair wiring. The interface uses 2 wires for transmission and 2 wires for receiving data, which allows this to be a full-duplex interface.

Transmitting data is called "Differential Voltage Transmission" because one of the transmit wires is negative, the other is positive. This assures that even with interference such as a change in voltage, this will affect both wires equally, and when their difference is compared if the voltage level between the 2 wires is greater than 200mV then the logic level is HIGH. If the voltage level between the 2 wires is less than 200mV then the logic level is LOW. Receiving data works in a similar manner.

MySQL Database Server:

We will be using MySQL 4.0.21.

The MySQL database is an open source relational database that is free under the GPL (General Public License), which allows modifications to be made to the code of the database. If a change is made and sold for profit, the change must be shared with the community. The source of the database server must be provided to the customer on request. We are not going to make any changes to the MySQL source and we shall refer them to www.mysql.com should they wish to edit the source.

Once the data is received on the RS232 serial line we interpret the data and enter it into the MySQL database.

The database will be basic during the prototyping phase. Each entry in the database will show: `ustomer_id`, `cell_ph_no`, `balance (in minutes)`, `time entered`, and `time left`.

Once the PC has the data of the complete transaction we have entries of the time when the car entered and left the spot. Using the information from the database we are able to subtract the two times and determine how long the car was parked in the space. For situations where car's park overnight, or past midnight we must also take into account the full date. When we calculate the time to bill the customer, the time of each transaction is rounded up to the nearest 10 minutes.

Apache Webserver:

We will be using Apache 2.0.52.

The webserver is responsible for serving the WWW browser clients. The webserver communicates with the database through the Perl (DataBase Interface) DBI to retrieve the information that the browser clients requested. This allows all the complex formation of a database query to be hidden from the customer, so they can just point and click.

WWW Browser:

We will be using Internet Explorer 6.0 as our main testing browser. Netscape should also work, but the layout of the website may not look as neat.

The WWW Browser clients are customers who would like to view open spots from the Internet, or check their account status, or make other queries to the database such as how long they have been parked in a spot.

The browser relays the request to the Apache Webserver through the HTTP protocol. We will also implement SSL (Secure Sockets Layer) encryption so that customer info is not readily available to hackers sniffing an Ethernet or Wireless connection.

Email / SMS Server: (if time permits)

We will set up an email server in order to send out notifications of parking expiry by cell phone. We will also send out account status by phone.

30 minutes before a car's time expires at a parking meter, we will send out an email to the user's cell phone if he/she has one, or would like to have that notification. Using Perl DBI we will run a script that will take the current time and subtract it from the time stored in the database. If that time, is the maximum time of the parking spot – 30 mins, then we send an email to the user's cell phone. In the process of sending an email over the phone network, it is translated to an SMS message which can be displayed on a phone.

Cell Phone Notification:

If a parking meter is expiring soon the user will receive a text message on his / her phone informing them of this. Text messages are a guaranteed delivery. Timing is another issue will have to be worked out. But most text messages arrive within 5 minutes of sending them, unless the customer's phone is off.