POZNAŃ UNIVERSITY OF TECHNOLOGY
INSTITUTE OF COMPUTING SCIENCE

DELPHI
Driving Tomorrow's Technology

CarTooth PROJECT - CAR PAYMENT SYSTEM
http://www.cs.put.poznan.pl/csidc/2002/

APPLICATION ID: 53SHTX

CarTooth PROJECT
CAR PAYMENT SYSTEM

Team Members Michał Kowalski
Bartosz Nowierski
Michał Rein
Łukasz Szajkowski

Project Mentor Jan Kniat, Ph. D.

Head of Department Jacek Błażewicz, Ph. D., Dr. Habil., Professor

1 Abstract

Paying for parking, gasoline or toll highways is always a waste of time and an inconvenient procedure. To solve this problem we introduce Car Payment System (*CPS*) which offers:

- full automation of paying for parking and other car-related payments,
- charging for actual time of parking in city parking zones,
- car monitoring.

Our solution will change life of every driver. They will not waste their time anymore searching for a parking meter because the parking meter will find them. They will not have to pay a parking ticket ever again since the devices of our system will pay parking fees for them automatically, not absorbing their attention. When they come back to their car it will be still there because our parking meter will keep thieves away. Moreover, drivers can easily pay, for example, toll road fees or gas charges without stepping out of a car.

While designing we intended to create a system which would be practical and fully economically justified. We put emphasis on ensuring transaction security and keeping the price of end-user devices low.

Car Payment System ideally fits into the idea of Personal Area Network. Every car has its own CarTooth with Bluetooth onboard. It exchanges information and makes transactions with stationary devices of our system using Bluetooth technology. These devices are scattered all over the city and create a set of wireless networks.

In this report we describe Car Payment System from many points of view. The most important part – CarTooth is discussed thoroughly with regard to hardware, software and related algorithms. We present other modules focusing on their functionality and explain how they cooperate with CarTooth. The report gives the detailed description of solutions to the problems concerning wireless safe transfer of money and recognition of the state of a car. We also cover requirements of users and marketing issues connected with putting CPS into practice.

2 System Overview

2.1 General Description

The goal of our project is to create a system which will make paying car fees easier. Amidst many possible usages of Car Payment System probably the most expected is facilitation of paying for parking. Therefore we are almost entirely focusing on this problem in the course of the project. But we keep in mind other cases to make the system universal and open. Our concept is presented in Figure 1.

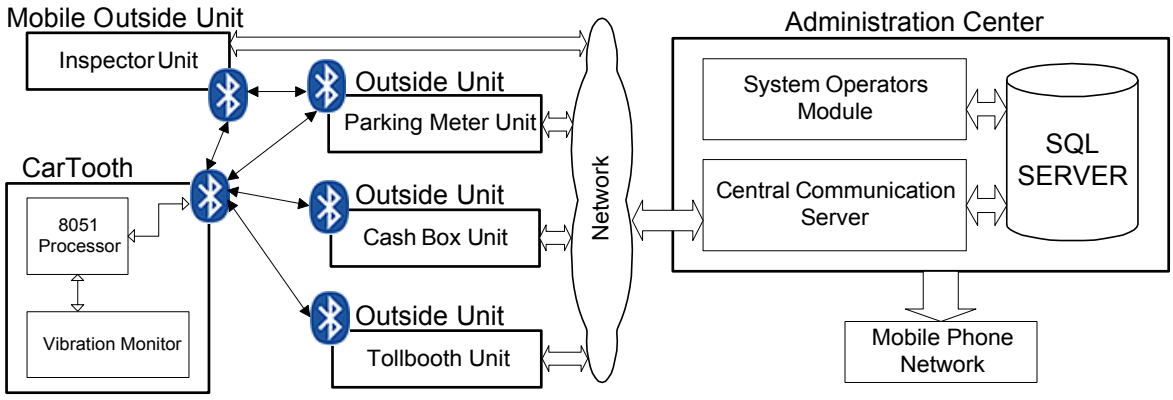

Figure 1. Overall system diagram

The key part of Car Payment System is CarTooth (*CT*). It implements an internal electronic account. Money from it can be spent by making transactions, via Bluetooth, with various stationary devices called Outside Units (*OU*). The account can be recharged in special OU – Cash Box Unit. It is not required to integrate CarTooth with a car, so there are no installation costs and it may be used in every car.

Paying parking fees in city parking zones can be realized in two ways (one automatic, the other semi-automatic). In the first case CarTooth contacts Parking Meter Unit (*PMU*) and makes periodic transactions. Together they are able to monitor a car and fire antitheft alarm when it is moved or vibrates suspiciously. The other way of paying must be used when there are no PMUs in a parking zone. In that case CT continuously decreases its internal account while parking. In order to check which cars pay their dues an inspector must come with his own unit (with

Bluetooth as well). He queries Parking Meter Units or CarTooths to retrieve a list of paying cars and give a ticket to other cars.

The system allows paying with CarTooth for any kind of product or service. For example, it can be used to pay for gas at gas stations, for a car wash or in drive-thru restaurants. Also when a car approaches a tollbooth on the highway its CT pays a fee and then the gate rises. Thus a driver can pass through without even stopping, not to mention looking for change or a special card.

Outside Units from all over the city are connected to the Administration Center (*AC*) by some kind of computer network (cable or wireless, e.g. GPRS). Its role is to supervise and coordinate the whole system. It authorizes, gathers information from Outside Units and sends managing data to them. The collected data is analyzed in order to detect frauds and create statistic for marketing use. AC is also responsible for notifying a driver (by sending SMS) or the police after receiving antitheft alarm. Thanks to the Administration Center it is possible to pay with the use of Parking Credit Account. In this case CarTooth works like a credit card – a client makes payments monthly with banking transfers instead of recharging the account using Cash Box Units.

2.2 Performance Requirements

Our system can be implemented in a wide range of hardware, but it has to meet some performance requirements. As a portable unit CarTooth should be small and light. It runs on batteries therefore low power consumption is very important. Moreover, its CPU must be able to cooperate with many internal modules.

Outside Unit is required to have enough computational power to serve CTs in a short time. Since establishing Bluetooth connection takes a few seconds, other operations, such as time-consuming authorization or making a transaction, should be done as fast as possible. On the other hand the multifunctional Administration Center needs computational resources to process a large amount of data (detailed calculation in point 3.4). It is also necessary to ensure an appropriate level of transaction security without lowering the speed of communication.

2.3 Design Methodology

In the process of creating Car Payment System we employed a mix of waterfall software development model and eXtreme Programming. First of all, we held interviews with potential users, clients and producers to learn market requirements our system should meet. Since we had no opportunity to have continuous contact with them we used a variant of Six Hats Technique (a kind of role playing) whenever difficult issues were to be solved. The information acquired during the interviews was essential in planning games we played. All that brought us to designing our system as a set of UML diagrams (with the use of Rational Rose).

The development process was divided into small iterations, which gave us an opportunity to control work progress and the process of achieving the design objectives as well. We made many spike solutions (small, informal experiments with ideas how to solve problems). It was very important since we planned to develop newly designed solutions and had to work with new technologies.

The testing process started simultaneously with development, but according to XP methodology test cases were created before any code was written. All parts were continuously tested. This all helped to detect errors early enough, when they were easy to locate and correct.

2.4 Innovative ideas

The main innovation is the full automation of car-related payments. Drivers no longer have to bother about troublesome paying for parking. They do not need to press any button since our system is able to detect when they park. Another ingenious feature, which makes paying for parking more justifiable for a customer (and thus makes the system more attractive) is monitoring. With no additional costs the system can guard clients' cars.

Our solution is open and flexible compared to other existing parking systems. CPS's multifunction ability allows coping with complex payment problems. Because of its good scalability, it can work in private parking lots as well as in whole agglomerations.

3 Implementation and Engineering Consideration

3.1 Users Requirements

Usefulness of Car Payment System is our prime objective thus we focus on a consumer. We have two kinds of users. The first group includes people who will run this system – employees of City Service like administrators, system operators or inspectors. We carried out several interviews. The director of Poznań Parking Zone helped us to recognize real needs of a potential customer. The representative of a potential system producer assured us that our ideas were good and worth implementing. The second type of users are drivers. We know their expectations well because all of us are drivers. We also questioned taxi drivers, deliverers and businessmen about their attitude to this project. All collected information highlighted the aims we should consider in designing.

3.1.1 Driver Use Case

Driver is a person who uses CarTooth in a car.

Paying parking fees. A driver uses this function when arriving at a parking place. There appears an announcement about the price of the place. The user can accept to pay or not. The fee is charged continuously until car departure.

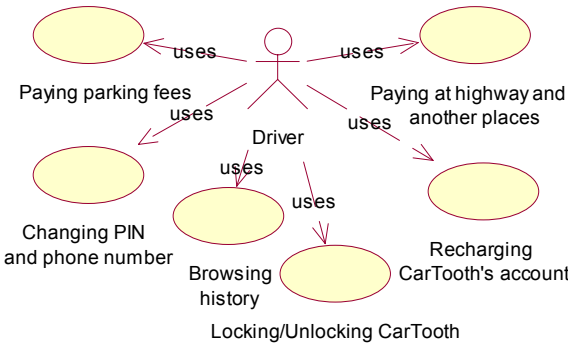

Figure 2. Driver use case

Paying at highway and other places. This functionality is common for many applications. When a driver approaches a service facility, information about it is displayed on CarTooth's screen. When the driver accepts to pay, a single transaction is carried out and the service can be used.

Locking CarTooth. A driver can choose from menu an option of locking a car, i.e. protecting it against stealing. Using a car would cause alarm, unless unlocked first.

All these 3 cases can be used either automatically or manually depending on user's preference.

Unlocking CarTooth. CarTooth can be unlocked by entering PIN.

Recharging. A driver uses this functionality when internal electronic account needs recharging in

a Cash Box Unit. If the user has Parking Credit Account in AC this function is disabled.

Changing PIN. A driver can change PIN number which protects CarTooth.

Changing phone no. A driver can change the number the information about the theft is sent to.

Browsing history. A driver is able to see time, amount and a place of ten previous transactions.

3.1.2 Inspector Use Case

Inspector is a person who traverses parking zones and checks if the drivers pay. This user carries Inspector Unit.

Viewing Parked Cars. Inspectors browse information on their devices about currently paying cars.

Giving a Ticket. This function is used when an inspector detects a car that is not paying. It is controlled manually.

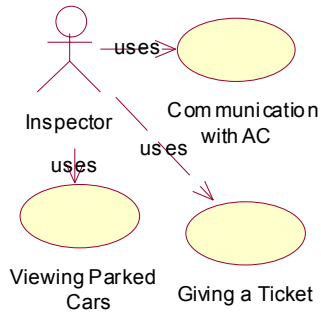

Figure 3. Inspector use case

Communication with AC. An inspector synchronizes data on Inspector Unit with AC.

3.1.3 Administrator and System Operator Use Case

Administrator is a person who manages a technical part of Car Payment System. *User Service* is used to create new user accounts in the system with appropriate rights. The Administrator enters information to CPS about new CarTooth and associates it with Parking Credit Accounts of driver. The Administrator uses *Statistic Service* to get information about load of the system and to get to know the traffic. System Operator manages services, which are offered to customers by CPS. In big systems, this position can be split according to Figure 5 into several posts, e.g. an accountant (*Drivers' Account Service*), an operator of parking zones (*Parking Zone Service*), an

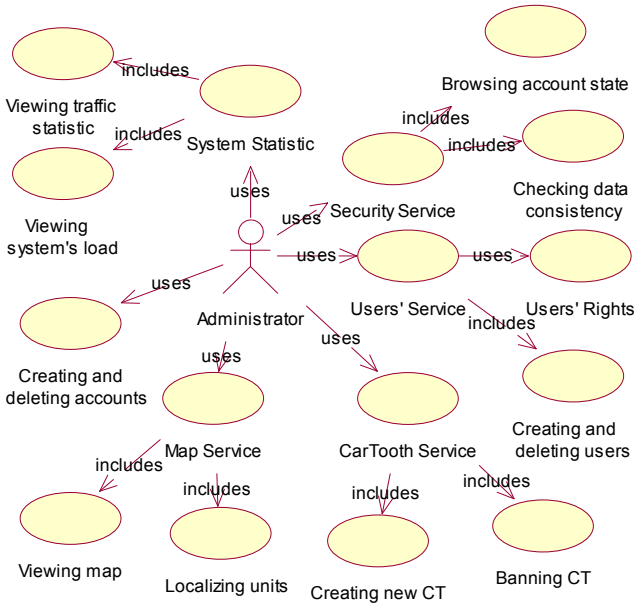

Figure 4. Administrator use case

OU maintenance technician (*Outside Unit Service*) and a CT operator (*CarTooth Service*). However, we have decided on joining all these functionalities in one position. It allows creating many multifunctional workplaces.

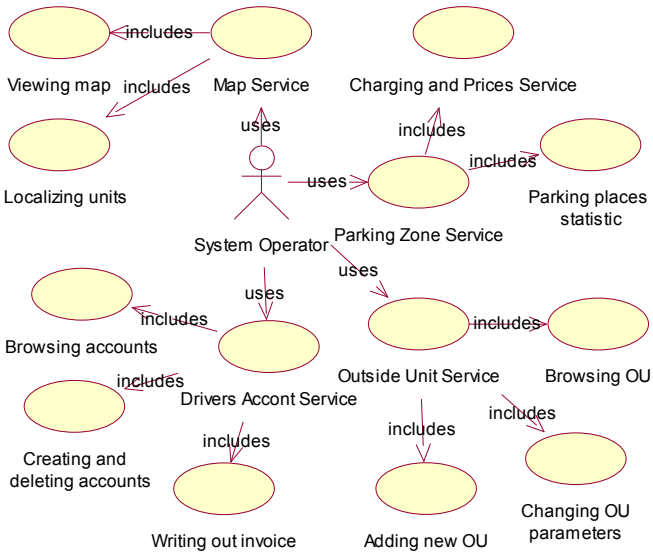

Figure 5. System Operator use case

Some functions are common to both System Operator and Administrator. They can find any Outside Unit or CarTooth on the map and browse *System Statistic*. These users can be also notified about frauds and antitheft alarms.

3.2 CarTooth (CT)

3.2.1 Functional Design

Users can set up CarTooth according to their preferences using the simple two-level menu.

Figure 6. CarTooth

Option	Description
LOCK ON/OFF	Forcing locking/unlocking of CarTooth
PARKING ON/OFF	Forcing starting/stopping paying for parking
LOCKING	Choosing a locking mode; AUTO – automatically locks a car if parked, MANUAL – prompts confirmation, OFF – does not ask about it
AUTOCHARGING	ON – automatically accepts transactions; OFF – asks for confirmation
HISTORY	Viewing time, place and cost of the last 10 transactions
REG. NUMBER	Viewing a car registration number
CHANGE PIN	Changing 4-digit PIN; user enters the old number and then a new one
PHONE NUMBER	Viewing/changing contact phone number (for sending alarming SMS)

Table 1. CarTooth's menu

Fully automation of CarTooth can be achieved when *AUTOCHARGING* option is set to *ON* and

LOCKING to *AUTO*. In such configuration CarTooth behaves as presented on the state diagram in Figure 7. Without these options the behavior is very similar, but CT asks some extra questions.

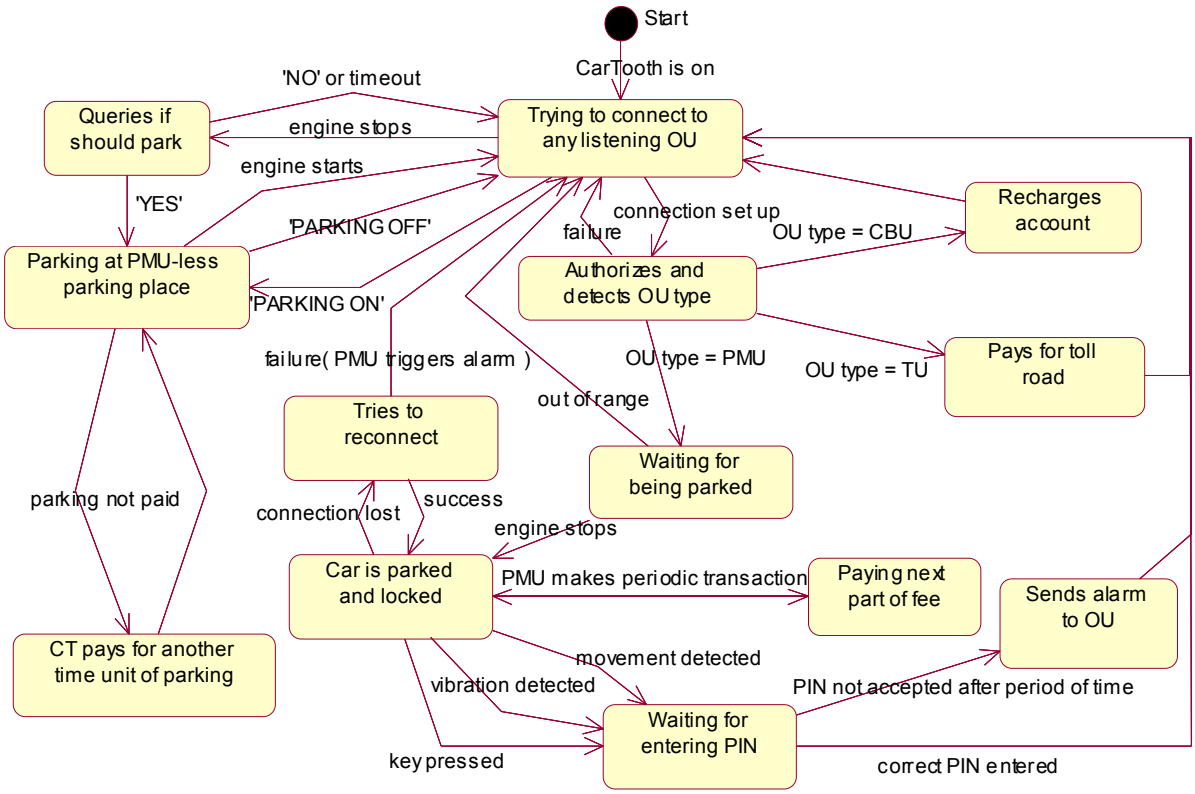

Figure 7. CarTooth state diagram

In PMU-less (i.e. without Parking Meter Units) parking places there is no Outside Unit to make transaction with. In that case every specified amount of time CarTooth makes a "self-transaction" – decreases its account and remembers total amount of money spent in such transactions. This value is later sent (and cleared) to the first Outside Unit that CarTooth contacts to.

Behavior of CT during cooperation with OU is explained in point 3.3.

3.2.2 Hardware Specification

A core chip of CarTooth is the popular inexpensive microcontroller AT89S8252 compatible with Intel's 8051 which is an industry standard. This IC has built-in 2kB EEPROM, a watchdog timer and capability of in-system programming. To interact with the user we use a standard LCD (4 lines 16, characters each) and a simple numeric keyboard (12 keys). Those devices allow us to build user-friendly, intuitive interface based on multilevel menus. Additionally, to notify the user of more important events which require his attention, we have equipped CT with a buzzer.

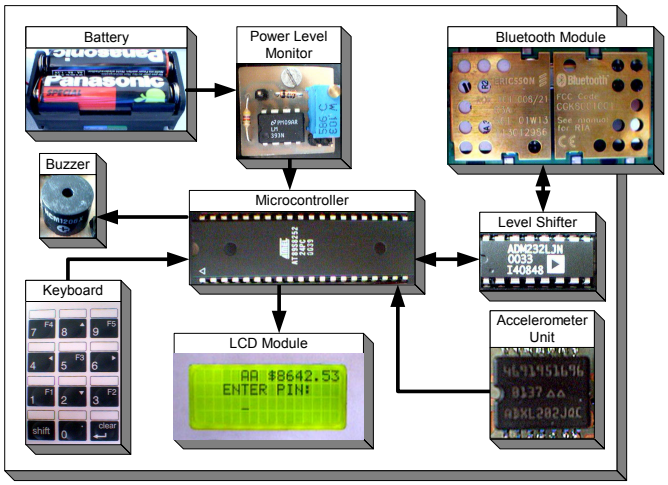

Figure 8. CarTooth block diagram

Because the received Bluetooth device has standard RS232 interface we had to add a level shifter to our device (TTL ↔ RS232). We have chosen the popular IC AD232 (compatible with MAX232) which allows transmitting data at the highest speed of our UART (115kbps).

In order to distinguish that a car has been parked and is not standing in a traffic jam we have to detect whether the engine is working. The basic guidelines are simple interface and low price of CarTooth, so we cannot use signals from car's electrical systems. The state of the engine is detected by analyzing vibration which is generated by every internal combustion engine. This vibration passes to the dashboard, and can be detected there. The IC used for vibration detection – ADXL202 is a very sensitive device that measures acceleration in two axes within the range of ± 2g. Information about acceleration is available in the form of a digital signal separate for each axis. It has constant, configurable cycle time and variable duty depending on the measured acceleration (0g corresponds to 50% duty). These signals are connected to blocking inputs of microcontroller's internal timers, which count t_X and t_Y (see Figure 9). So the CPU does not take part in measuring. It only has to read the result of measurements and reset the counters. This device serves also as a detector of stealing a car or removing CarTooth.

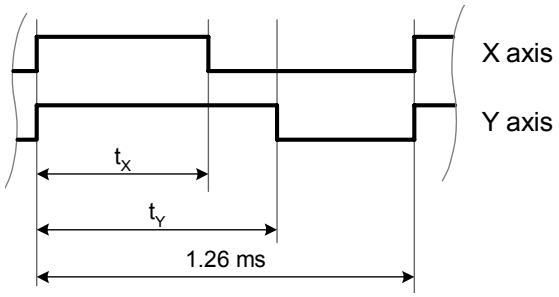

Figure 9. Accelerometer output

3.2.3 Engine State Detection Algorithm

The aim of the algorithm is to detect vibration coming from a working engine and distinguish it from another type of vibration. This problem can be reduced to checking if frequency of a signal

taken from the accelerometer is high. But there is one basic difficulty. We cannot rely on any level of reference ("0" level) of the signal, as gravity is always affecting it differently (depending on the angle of inclination). We introduce the algorithm which solves this problem.

1. Reset frequency variable *F*.
2. Repeat steps 3 to 6 for 256 readings of the signal (about 0.25 seconds) and then go to step 7.
3. Keep the maximum value of the signal. When it grows update the maximum.
4. When the signal falls check if it drops below the maximum more than the specified threshold. If yes then increase *F* and go to step 5; if no and it starts growing then go to 3.
5. Keep the minimum value of the signal. When it falls update the minimum.
6. When the signal grows check if it rises above the minimum more than the specified threshold. If yes then increase *F* and go to step 3; if no and it starts falling then go to 5.
7. If *F* is high (higher than the specified frequency threshold) then probably the engine is on.
8. To make sure repeat 20 times steps 1 to 7 (about 5s altogether) and if the positive answer is given in at least 16 of the cases then with very high probability the engine is working.

While creating the algorithm we carried out several experiments to establish the best possible values for the mentioned thresholds and optimal number of readings of the signals.

3.2.4 Movement Detection Algorithm

This algorithm is to detect movement (either displacement or turn). The only thing that needs to be done is to integrate acceleration to get velocity. The problem of the lack of "0" level is still present. Without knowing it we cannot calculate the integral correctly. Fortunately, if CarTooth is not moved then this level is constant; thus integrals taken from the same number of accelerometer readings should have (more or less) the same value. So we propose the algorithm as follows:

1. Sum 256 consecutive readings of the signal from the accelerometer (about 0.25 seconds).
2. If the calculated sum differs from the previous one (unless it is the first one) more than the specified threshold then probably CarTooth has been moved.

3. To make sure repeat 5 times steps 1 and 2 (about 1s altogether) and if the positive answer is given in at least 4 of the cases then with very high probability the device has been moved.

This algorithm is applied to signals from both axes. It is sufficient when at least in one case the movement is detected. The same applies to the algorithm from the previous point.

3.2.5 Tradeoffs and Optimizations

In order to enable rapid development of minimum resources consuming software for AT89S8252 we have developed a quasi-object mechanism. The characteristic of 8051 CPU does not allow implementing objects in a classic form. The power of this mechanism allows us to select a new active quasi-object by modifying the contents of one register, which changes the device responses to events. This permits us to alter the existing menus and add new levels of menus in a very simple and quick way without writing the whole new code for it. Also it simplifies the main loop – it is only responsible for calling a method of an active object in response to an event.

CarTooth has to work for a long period of time on battery power supply. Therefore one of the main problems is power consumption. Although Bluetooth needs 5V supply we have decided to use ICs which can work either at 3.3V or 5V. This would allow us to lower the voltage when using another Bluetooth module which will result in lowering power consumption. Because of these constraints we have chosen a microcontroller with idle and power down modes and equipped CT with software control of the most power-consuming element – LCD back-light.

We also wanted CarTooth to be handy and small. That is why we used the microcontroller with built-in EEPROM and why our program uses only built-in 256 bytes of RAM. This allowed us to use one chip instead of four big ICs. Surface mounting allowed us to make even a smaller device. From many accelerometers we have chosen one with digital output because it can be connected directly to the microcontroller without any power-consuming and place-taking AD converters. The microcontroller's internal-timer blocking inputs are also interrupt inputs so there are no problems with synchronization. But the signal's cycle starts with high level and the moment

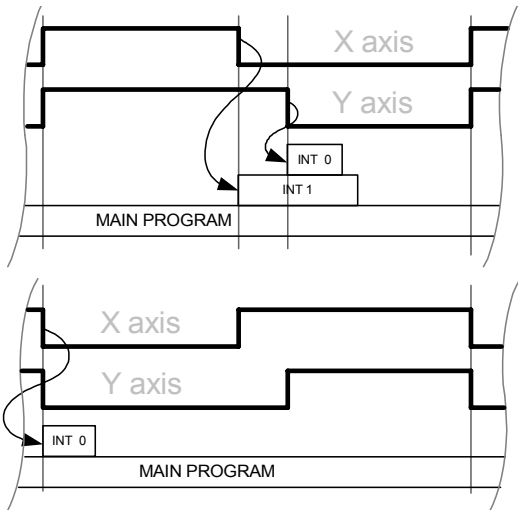

Figure 10. Interrupts

when falling edge occurs corresponds to acceleration value. The interrupt subroutine is called on the falling edge so if the edges were not synchronized we would need two interrupts (see Figure 10). That would complicate the program and use more time to read a single measurement. That is why we had to complement signals from the accelerometer (using 74LS05 inverter) – now the falling edge on both lines occurs in the same moment so one interrupt is enough to read both values.

Next objective that caused troubles was a statement about low price of CarTooth. We were thinking about LCD touch-screen but it is much more expensive than a typical set of LCD and keyboard. Also when choosing a method of unlocking a car we had to make compromises to fit in our aims. We wanted to construct a user-friendly device so we did not want to burden users' memory with PIN. We have even started to construct a system allowing authorization of a user based on comparing fingerprints. But even using a relatively cheap scanning device we would double the cost of CarTooth and we would have to use more powerful CPU to process data from the scanner. That would cause the increase in power consumption and necessity of increasing battery's capacity which causes enlargement of CT device. To sum up – big troubles with only little increase in functionality. That is why we decided to use a typical way of protection applied in cash machines and cellular phones – PIN. Potential clients are accustomed to this kind of protection so they should not have any trouble with unlocking. In order to make entering PIN easier we equipped CarTooth with numeric keyboard. This is also helpful in navigating the multilevel menu and in entering other alphanumerical data such as a telephone number.

3.2.6 Testing

First of all we tested how Bluetooth module communicates from within a car. It turned out that its

range narrowed to about ¾ of nominal (it is still good enough). The next tests were experiments that checked out our vibration detection theory. In order to do that we have built a simple device that transmitted data from the accelerometer using RS232 port. We have recorded data from a car in motion, parked car, parked car with a working engine (in cars of different classes) and vibration caused by a tram or a truck passing by or even caused by hooligans trembling a car. For further testing we have implemented our algorithms on AT89S8252 and wrote a program displaying the results of analysis on LCD. The effectiveness of the algorithms was satisfying.

3.3 Outside Unit (OU)

All devices that are outside the Administration Center and are mainly designed for selling different services and goods are called Outside Units. OU communicates with CarTooths using Bluetooth technology. It also communicates with Central Communication Server (*CCS*) using commutated links. Communication and security algorithms are described in points 3.7 and 3.8.

Outside Unit buffers transactions made with CarTooths. It also fetches data from CTs about payments made in PMU-less parking places and creates artificial transactions from this information (and put them into the buffer as well). Every specified time or when the buffer is going to overflow Outside Unit tries to connect with CCS to send the data and flush the buffer. Connection can be also established on demand, e.g. when Parking Meter Unit wants to fire antitheft alarm. Occasionally, Outside Unit exchanges additional information with CCS. It sends results of self-tests, receives managing data and lists of banned CarTooths.

Since connection with Central Communication Server is implemented using a standard protocol, the physical connection can be set in various ways; for example, Outside Units can create small networks or scatternets where only one of OUs has a direct connection with CCS (cable or wireless). It can be done using Bluetooth technology or any other. There is another interesting possibility. When OU does not have any medium to connect to CCS, the connection can be established via any portable device equipped with Bluetooth and, for example, GSM interface.

Extended Inspector Units can play this role. This solution does not decrease the level of security.

Because Outside Units are placed in public places lots of Bluetooth modules not belonging to our system may appear in their range. Thus pre-selection and ignoring of devices, both implemented in our communication protocols, are very important for OU efficiency.

3.3.1 Parking Meter Unit (PMU)

Parking Meter Unit is mainly responsible for charging fees in city parking zones. When CarTooth enters the range of one of PMUs, the connection is established. Since then, information can be exchanged and transactions made. Parking Meter Unit has additional ability of monitoring cars. Antitheft alarm is sent to the Administration Center when CarTooth is locked and one of two conditions is met. The first is when CT informs about the theft. The other is when CT goes beyond PMU range. If the second condition is met alarm is fired instantly but it is postponed by AC, giving the suspected CarTooth some time to cancel it (to avoid accidental alarms).

We have considered another solution for the mentioned problem, as a tradeoff. There is an option that, instead of the Administration Center, Parking Meter Unit postpones broadcasting the alarm. CarTooth has to reconnect to the same PMU to cancel it. This solution is easier to implement and decreases a required number of connections to AC. But there is a problem when CT accidentally connects to the farthest Parking Meter Unit, on the border of its range. Such a connection can be easily lost when external conditions change (e.g. a truck parks between CT and PMU). Therefore there should be a possibility of connecting to another PMU. That is why we have decided on the solution from the previous paragraph.

3.3.2 Inspector Unit (IU)

Inspector Unit is a special kind of Outside Unit. It is mobile and can cooperate besides CarTooths with Parking Meter Units. It should be small and handy, e.g. a palmtop. It is carried by an inspector. Inspector Unit collects information about devices that are paying for parking. Registration numbers from such devices are retrieved and listed on its screen. Cars not listed are

not paying and should be given a ticket.

In PMU-less parking places Inspector Unit works like every Outside Unit. It accepts connections from all CarTooths within its range and then queries them in order to learn if they pay. In parking places with PMUs Inspector Unit connects to them (details are described in points 3.7.3 and 3.8.2) and retrieves information about paying cars.

3.3.3 Cash Box Unit (CBU)

Cash Box Unit is used for recharging CarTooth's accounts. To do that CT must be connected with CBU. Drivers choose their device from the list of attached CTs presented on CBU's screen. Then they must pay (using credit card or cash – depending on implementation) the amount they want. For better protection against frauds CBU must be connected with AC to make a transaction.

3.3.4 Tollbooth Unit (TU)

Tollbooth Unit is created mainly to charge for toll roads. There are two major problems we had to consider while designing this unit. The first is that if more than one CarTooth is in TU's range it has to decide which CT should be billed. We have thought up that simple video camera can solve it. By taking a picture and using OCR, TU is able to recognize a registration number of the first car in a queue. This information is enough to associate the car with CarTooth, because every CT remembers registration number of a car it is in. The other problem arises when every Tollbooth Unit controls only one gate, then CT might connect with an inappropriate one. It shall be solved by connecting TUs into a network. These solutions give a simple scenario of paying at tollbooths: a car slows down, CarTooth connects with TU, it is recognized and billed, a gate opens and a car gets through. Usually establishing Bluetooth connection, authorization and OCR take about 6 seconds so a car may drive up to 6mph to get through a gate without stopping.

3.3.5 Other Possible Extensions

There can be lots of types of Outside Units. For example modification of Tollbooth Unit can be used in parking lots. Additionally we should only assure that a car is associated with CarTooth

when entering (mainly for a purpose of securing cars) and billed when leaving. The solutions proposed in TU can be also used for a wide range of payments, e.g. at gas stations or drive-thrus.

3.3.6 Testing

We implemented and tested a universal Outside Unit that communicated with CarTooths and sent data to Central Communication Server. After that we started implementing specific types of OUs (described above) basing on the universal one. This process significantly reduced a possibility of errors in implementing specialized OUs. We tested them in complex practical situations.

3.4 Administration Center (AC)

The Administration Center is a set of modules that gives ability to manage and coordinate the whole system. The core of AC is SQL database server. It stores information about users, devices and all events that occur in the system. Central Communication Server (*CCS*) is a database server's interface for the devices of our system. It authorizes, gathers information from Outside Units and sends managing data to them. It can automatically detect frauds and broadcast antitheft alarms. System Operators Module (*SOM*) is the program which allows operators to browse, add and modify data. It also makes advanced analysis of information.

The Administration Center has to process lots of data. On the basis of information acquired from the interviews we can estimate that in a big city of about 10 million inhabitants there are up to 150 thousand paid parking places. Dividing it by 5 (average number of parking places per Parking Meter Unit) gives 30 000 Outside Units. Assuming the expansion of our system into parking lots, gas stations or drive-thrus we must add about 15 000 OUs. That makes about 45 000 of OUs. Assuming that an average OU makes a transaction every 6 minutes 10 hours a day (periodic transactions made between PMU and CarTooth are cumulated) it gives 100 transactions per day. The record for one transaction is 48 bytes long. That all gives 45 000 * 100 * 48 = 216 000 000 bytes (about 206MB) a day. This shows that SQL database server is very important and should be chosen with deliberation. We will use this calculation in the following points.

3.4.1 Tradeoffs

While designing we had to choose a type of communication between Central Communication Server and System Operators Module. There was an alternative of direct TCP/IP connection or connection through database. In the first option CCS would have to buffer orders, but SOM users would have ability of more interactive communication with OUs. In the second one there is no need to establish additional connections, because both SOM and CCS are connected with the database server. Additionally SQL server buffers orders which exempt other modules of AC from doing it. We have chosen the solution with connection via database, because it gives enough functionality, is easier to implement and protects buffered orders using SQL server mechanisms.

Another considered problem was AC's reaction to antitheft alarm received from OU. There was an alternative of automatic broadcasting by CCS or manual one made by operators using SOM. The first solution was chosen, although it does not provide an option of filtering alarms, which could limit a number of false ones. The advantages are that there is no need of continuous work of additional operators and there is no possibility of identifying real alarm as false. We think that this solution is good since our theft detection algorithms and procedures have high credibility.

3.5 Central Communication Server (CCS)

Central Communication Server is an interface between Outside Units and SQL database server. It has also modules for detecting frauds and for broadcasting antitheft alarm. Excluding these two modules CCS is a typical communication server. It translates data from our format into SQL queries and sends those to the database server. Broadcasting antitheft alarm is not yet implemented because its implementation depends on an interface to a mobile phone network. We have made C++ objects that emulate broadcasting therefore it will be easy to include real mechanisms in the future. Now antitheft alarm is shown on the screen.

3.5.1 Fraud Detection Algorithm

This algorithm is used to detect deceptions made by CarTooths, e.g. when they did not decrease

their accounts or pretend they did not make the transactions they did. The algorithm bases on transaction entries in the database, which consist of: *OU_id*, *OU_transaction_number*, *OU_time_begin*, *OU_time_end*, *CT_id*, *CT_transaction_number*, *CT_account_before*, *CT_account_after*, *CAR_registration_number*.

Whenever new data comes do for each *CT_id* from new entries the following:

1. Take the last certain transaction and a new one with the lowest *CT_transaction_number*. Let the certain transaction have *CT_transaction_number* equal to *X* and the new transaction to *Y*.
2. If *Y* is not greater than *X* go to step 6.
3. If *Y* is greater than *X+1* look at *OU_time_begin* of the *Y* transaction. If every OU has uploaded its data after that time go to 6 or else stop.
4. Check if *OU_time_begin* of *Y* is greater than *OU_time_end* of *X* and if *CT_account_before* of *Y* is equal to *CT_account_after* of *X*. If any of these conditions is false then go to 6.
5. Mark *Y* as the last certain transaction. If there is still any uncertain one go to 1, else stop.
6. CT cheats and must be banned. This information is stored in the database and broadcasted via CCS to all Outside Units. The owner is asked for explanation.

3.5.2 Testing

Besides making other tests (see point 3.9), we tested this part by establishing communication between OUs and SQL server through CCS. For fraud detection algorithm, besides testing it, we have proven its correctness. We tested Central Communication Server's performance. Assuming it would have 10Mbps connection with the network (PVN or internet) and all OUs would try to send lists of transactions at the same time, it would have about 4 minutes to cope with 206MB of data (see point 3.4). Our tests showed that CCS has no problem to do that even on a standard PC.

3.5.3 Tradeoffs

The most important tradeoff considered during designing CCS was whether to buffer data from OUs or not. The first solution optimizes time of connection because CCS can store data in RAM,

send confirmation to OU and transfer it later when SQL server is idle. The other solution is slower, since OU has to wait till CCS successfully sends data to database. We have chosen the latter option because it is much more secure. Good SQL servers implement data recovery procedures in case of a crash. It would not be reasonable to implement them in CCS, but OU has to be sure that the data sent is permanently stored, because it deletes its copy after CCS confirms.

3.6 System Operators Module (SOM)

This part of the Administration Center offers visual access to the functionality available for system authority. Each module implements some use case from Administrator and Operator use case diagrams (see 3.1.3). SOM makes managing Car Payment System easier and it adds a possibility of creating strategy for possible expansion of the system. *OU Manager* can add new Outside Units and maintain already existing ones. Users of the system are able to see parameters of these devices such as level of power, operation history and others. Similarly, *CT Manager* can add a new CarTooth and has an access to its parameters. Some CTs are associated with Parking Credit Account. It can be browsed, changed and created in *Account Manager*. This part of SOM writes out invoices to companies. It also controls financial transactions in our system. *Security Manager* is responsible for safety issues such as managing public keys and informing System Operator about frauds or antitheft alarms. In this way we guard CPS against hacking and lower a possibility of forging our devices. Operators and Administrators can easily localize OUs and CTs by using *Map Service* (for CTs the last known position is shown). Additionally this part assists *Parking Zone Manager* in maintaining localization, prices and other parameters of parking zones. Users can apply *Statistic Manager* to monitor information about level of the traffic or load of CPS. Moreover, it is possible

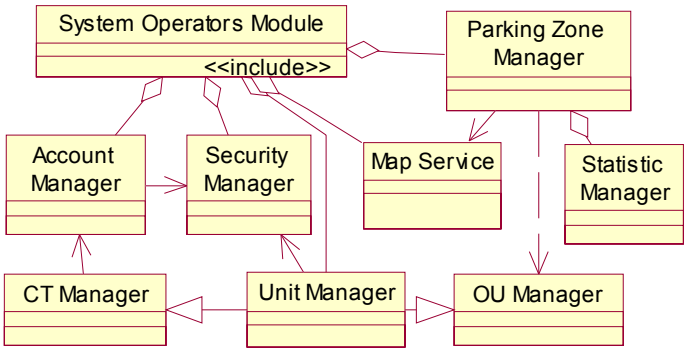

Figure 11. System Operators Module structure

to create profitability statistic. Basing on this data some marketing decisions can be made.

In this part of our project we have to solve the problem of access of many independent managers to the database. The easiest way is to provide each submodule with separate connection to the SQL server. This idea guarantees fast access to data and quick flow of information. *Unit Manager* is a different solution. It forces all managers to store and load data from a local buffer. Then these records are exchanged with the database server. *Unit Manager* thus controls data consistency at System Operators Module level. It accepts changes and mediates between SOM and the database server. Although this solution makes refreshing of information more inert, we have chosen it to provide consistency of data. It is comfortable for users as they do not have to worry about propagation of changes.

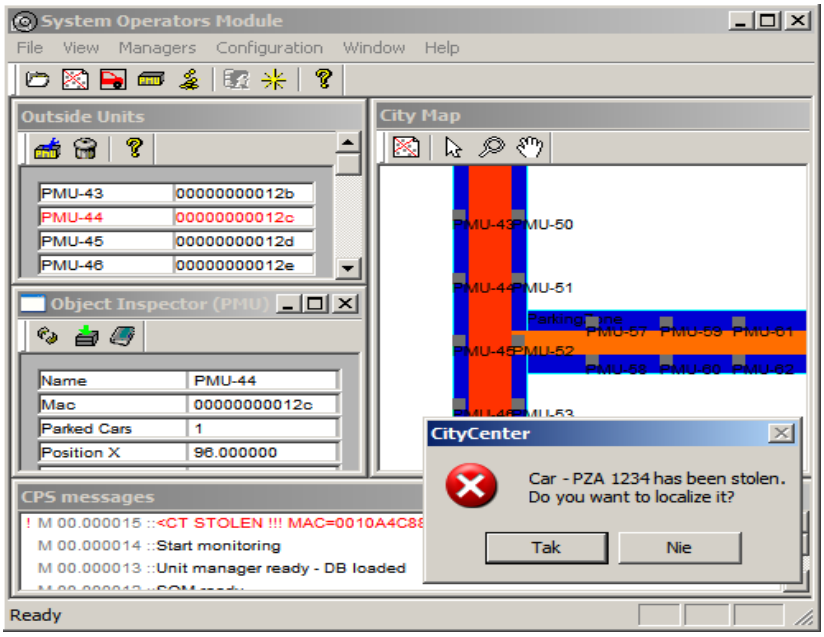

Figure 12. SOM screenshot

3.7 Communication Protocols

3.7.1 Communication Interface

To contact with a Bluetooth module we use Host Controller Interface (HCI). Unfortunately HCI requires much work from a programmer and we could not afford such work for every kind of units. So we have developed a communication library, which is common for every device which uses Bluetooth, except CarTooth where we could implement only a small part of it. Its strength is that many HCI events and commands are processed in background, which allows a host to cooperate with Host Controller without affecting the flow of application. Additionally, the interface of our library is very similar to BSD sockets. It is a great advantage as this interface is

much more user friendly than HCI and is well known for every network programmer. Now writing a Bluetooth application resembles writing a client-server application.

In our library most functions have the same meaning as in original sockets, some may have decreased abilities. It is worth mentioning that we have implemented both a packet and stream type of communication. Moreover, I/O can be used either in a blocking or non-blocking manner (thanks to *select* function). We have added additional functionality to support some Bluetooth features. For example: a client does not have to specify a server's address (it may connect to any in the range); a server may reject Bluetooths with another class of device; units with specified hardware address can be ignored. All this makes our library universal, so it can be used in any kind of application that uses basic Bluetooth functionality and it can be easily expanded to support more complex functionality.

3.7.2 Outside Unit – CarTooth Protocol

In contact between OU and CT we use sockets with communication of a packet type. The packets can have up to 117 bytes to fit in ACL DM3 packet (for our application we need only 64B).

Just after establishing the connection both units do some initial, security related procedures (see point 3.8.1 for details). Further communication protocol bases on packets presented in Figure 13. Every packet is either a command, a request for some data or a reply. *Type* field determines what it is exactly (for the list of types see Table 2), whereas *Data* is a variable length field for parameters preceded by *Data size*. Then the packet is padded with zeros so that its length is a multiple of 16 bytes and encrypted with AES.

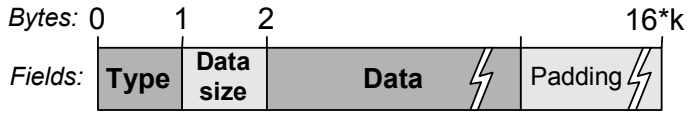

Figure 13. CT-OU packet structure

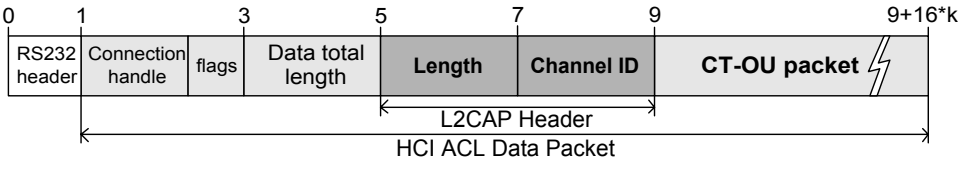

Figure 14. Data package structure

The CT-OU packet, before sending to Host Controller, is equipped with L2CAP header,

encapsulated in HCI ACL Data Packet and then preceded by HCI RS232 Packet Header (as shown in Figure 14). This is all done in background by the socket library.

Type	Description	Parameters
cmdTransact	Tell CT to make a transaction (update an account)	Amount (– or +)
cmdDisplay	Tell CT to display a text on its screen	Text to display
cmdQuestion	Tell CT to display a question on its screen and to send back an answer	Question to display
cmdSetLock	Tell OU (PMU only) to lock or unlock a car	'ON' or 'OFF'
cmdAlarm	Tell OU (PMU only) to trigger alarm	
reqRegNo	Request from CT registration number of a car	
reqAccount	Request CT's account state	
reqTransNo	Request the last transaction number made by CT	
reqPMUlessT	Request from CT amount of PMU-less transactions	
reqPMUlessS	Request from CT current PMU-less paying status	
repAck	Acknowledge that the command was received and the required action is done	Type of the command
repData	Send the requested data or answer to the question	Type of the request or command followed by data

Table 2. Packet types

3.7.3 Parking Meter Unit – Inspector Unit Protocol

In contact between IU and PMU we use sockets with stream type communication. This type is very comfortable, because from the point of view of a higher level protocol data of any length can be sent in one piece. Of course the data is not moved to Host Controller in such a form. It is first fragmented into smaller packets (not longer than 117 bytes to fit into DM3 packets). Each of them is encapsulated similarly as in Figure 14 and then sent to HC.

After connecting the units do some security procedures (see 3.8.2 for details). Then PMU sends a list of registration numbers of all cars that pay a parking fee. These numbers are sent in ASCII standard. They are separated by `0Ah` ('\n') character and the whole message ends with `00h`.

3.7.4 Testing

To test the Bluetooth socket library we employed a white-box technique. We wrote several simple socket applications and created test cases that guaranteed covering all lines of code. These tests (and tests described in point 3.9) made us sure that the code is reliable.

3.8 Security algorithms

One of our main objectives was to create a secure system. Therefore we have decided to give up using Bluetooth security techniques and to introduce stronger ones. For ciphering we use the newest standard – AES (Advanced Encryption Standard, a.k.a. Rijndael) with 128-bit key, for 128-bit data blocks. In case of authorization we base on RSA with 512-bit long keys.

3.8.1 Outside Unit – CarTooth Communication

For authorization every CarTooth has in its internal memory an important value A_{CT}. This is the hardware address (MAC) of CT's Bluetooth encrypted with RSA algorithm using a private key D known only to the producer. The decryption key E (complementary to D) is public, thus known to every Outside Unit. Additionally, every CT and OU has got 128-bit key K_0 (also in their internal memory).

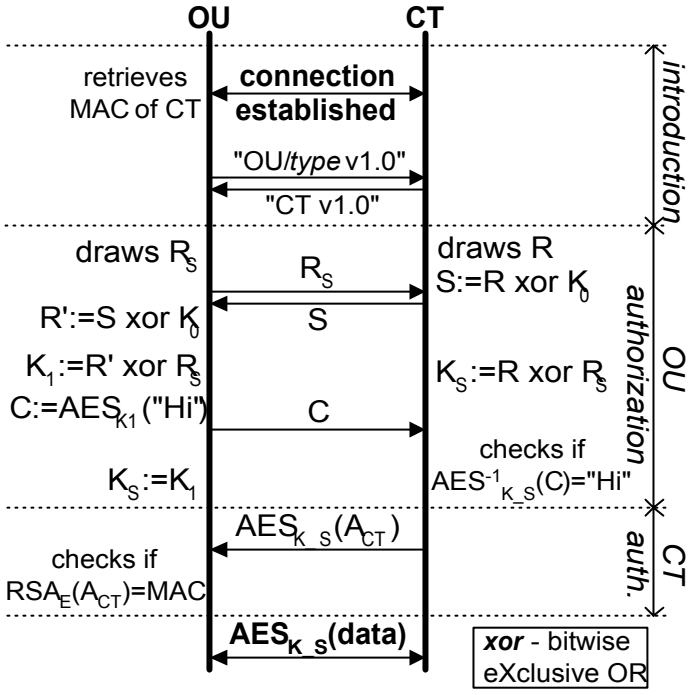

Figure 15. OU-CT handshaking

Authorization of both units and key negotiation is presented in Figure 15.

3.8.2 Parking Meter Unit – Inspector Unit Communication

Inspector Unit does not have to authorize as it gets only a list of paying cars. But a stronger authorization of Parking Meter Unit is required. For this reason every PMU has its own private

key D_{OU} for RSA algorithm (every OU has such a key and uses it to authorize to the Administration Center as well). Each Inspector Unit keeps the list of public keys E_{OU}. The authorization of PMU is described in Figure 16.

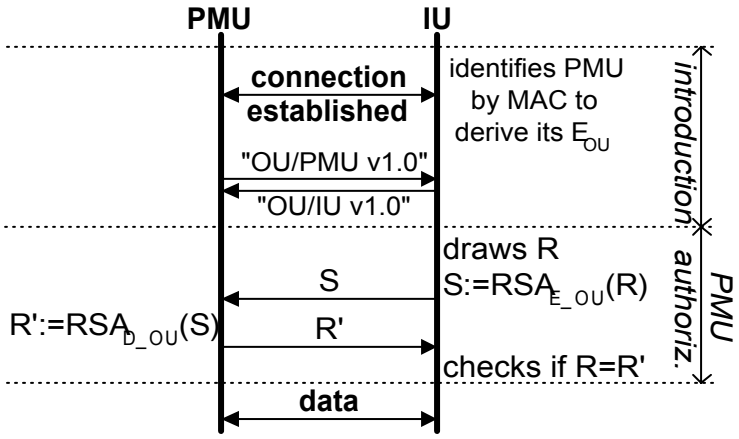

Figure 16. PMU-IU handshaking

3.8.3 Outside Unit – Administration Center Communication

Authorization of AC and OU is based on RSA and MD5 standards. The Administration Center has got a private key D_{AC} and a public key E_{AC} known to every unit. Every Outside Unit has its own private key D_{OU} and a public key E_{OU}. AC keeps the list of all E_{OU}s. Authorization and key negotiation protocol is shown in Figure 17.

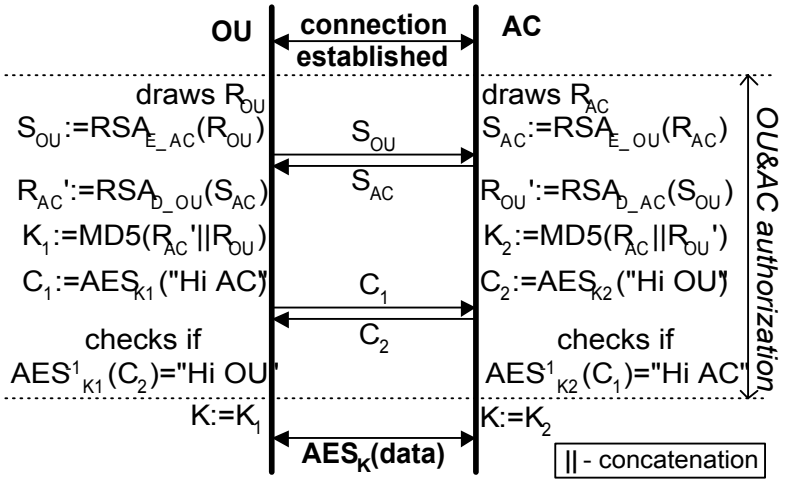

Figure 17. OU-AC handshaking

3.8.4 Tradeoffs

We wanted to make Car Payment System as much secure as possible. Unfortunately it is in contrast with the assumption of simplicity and low cost of our system. Therefore we were forced to implement weaker security algorithms, but still we have managed to protect our system well.

The most important thing for us was to prevent CarTooths from being forged, because it could cause great financial losses to an owner of CPS. It would be the best if CT could authorize to the Administration Center in the way presented in point 3.8.3. But it would require Outside Units to connect to AC for every transaction and CT to have a better processor to cope with RSA in a

short time. Such a solution would be too expensive to create and to use according to our assumptions. Nevertheless our proposal is a good compromise as it has a few levels of protection against frauds. Even if A_{CT} number is somehow intercepted (which is very difficult) it can be used to create a fake CarTooth which must have Bluetooth with the same hardware address. It is not easy, if possible at all in home conditions. But even if it can be forged the frauds will be easily detected by AC in a short time and such a device will be banned, thus unusable. To conclude, forging only one CarTooth would take a lot of time and money, while profits would be low.

Authorization of Outside Units presented in point 3.8.1 is not ideal. It is based on a common key K_0. It would be much better if every OU had its own key. But in that case every CarTooth would have to keep a list of all OUs' keys. This is unacceptable in our system. The solution we have proposed is good enough since retrieving keys from internal memory of devices is extremely difficult and expensive. Moreover, there is no point in forging OUs, because it will not give any profit at all. But even if someone made such units they can be detected easily by inspectors.

3.8.5 Testing

Testing security algorithms we have implemented was relatively easy. RSA, AES are MD5 common standards, thus there are lots of materials about them available. Among them we have found exemplary inputs and outputs for all these standards, so we have tested our software on it. Additionally, to be entirely sure, we have applied thousands of random tests in which we encrypted random data, then decrypted it and checked if we got data before encryption.

3.9 Testing Summary

The testing process started simultaneously with the development. We made many spike solutions. Developed parts were continuously examined in isolated tests. That helped to detect errors early enough when they were still easy to locate and correct. Tests were mainly made using black-box technique. It concentrates on desirable behavior and allows constructing tests before development. All modules were integrated as frequently as possible, which decreased the risk of

incompatibility of modules. The specific tests for all parts are described in appropriate points. Our tests have proven that the part of the system we have implemented works correctly.

3.10 Developed Tools

In the process of developing CPS we have built some useful tools.

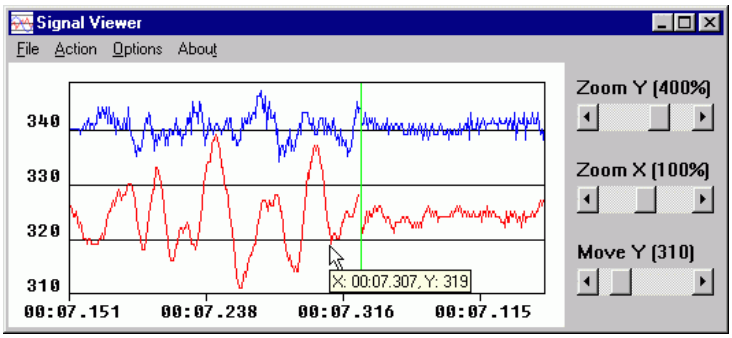

Figure 18. Signal Viewer screenshot

Signal Viewer is the application that presents a signal flow on a chart. Consecutive samples of the signal can be read either from a file or the standard input. It can be viewed at different speeds, step by step or paused. Zooming and scrolling a chart is available too. *Signal Viewer* was essential while developing the vibration and movement algorithms. First we used it to observe an accelerometer signal in order to understand it better. After implementing the algorithms this application helped us set up parameters experimentally for them and debug them.

To avoid many program uploads to the CPU (durability of its flash is limited) we mounted hardware ROM emulator and evaluation board for ’51 with additional ICs. This allowed us to develop the software quicker. We also built a simple programmer, which uses port for CPU’s in-system programming.

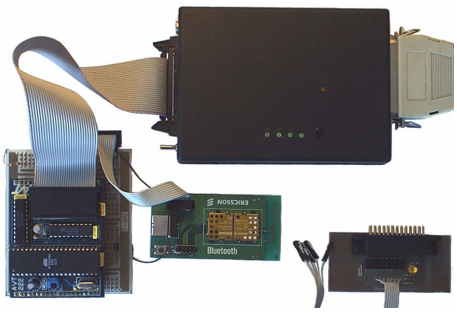

Figure 19. Hardware tools

3.11 Marketing Outline

We have considered the cost of putting Car Payment System into practice for drivers and for authorities. Drivers will buy or rent CarTooth device willingly. It offers an extra protection and it cost very low. Especially for companies we have created an option of Parking Credit Accounts. They would use it to pay monthly with transfers and to simplify inclusion of parking fees to the operation costs. For city authorities CPS is more profitable than other solutions. Usually cities have a parking system installed or cannot afford single big expenses. Therefore we have made our system modular so that putting it into practice can be divided into some stages. From the very

beginning Car Payment System starts bringing profits whereas the biggest costs appear at the end.

I. Initial stage. Drivers get CarTooths, inspectors are equipped with Inspector Units. There is no need to buy Parking Meter Units. Cash Box Units are necessary but only a few are required. Therefore the cost for authorities is very low. Drives are charged for actual time of parking, but automation and car monitoring are not available yet.

II. Transitional stage. Authorities begin to install Parking Meter Units in the most loaded zones, which makes automation and car monitoring. It can coexist with the previous parking system and PMU-less parking places. Moreover, other payment units make using CarTooth more attractive. Although this stage is more costly, profits become much higher.

III. Ultimate stage. At this stage there are Outside Units installed in the whole city. Finally CPS has full functionality. Authorities have high profits because CarTooth is a device of everyday use.

We have compared the cost of Parking Meter Units to the devices that are operating in Poznań. In our home city there are Siemens units. One device costs about $10,000 and serves 24 cars, so one parking place costs **$416**. Our PMU will cost about $350 but it serves averagely 5 cars, so for one car it would be **$70**. This calculation shows that the cost of our system is very low.

3.12 Additional Costs

Name	Quantity	Price per piece	Cost
ADXL202 (accelerometer)	1	$19.95	$19.95
LCD	1	$10.79	$10.79
Keyboard	1	$2.50	$2.50
AT89S8252 (microcontroller)	1	$3.00	$3.00
ADM232L (level shifter)	1	$1.25	$1.25
BC517 (Darlington transistor)	2	$0.05	$0.10
74LS05/SO (inverter)	1	$0.10	$0.10
LM393N (comparator)	1	$0.10	$0.10
Capacitors, resistors, PCB, buzzer, etc.	-	-	$7.50
		TOTAL :	**$45.29**

Table 3. Additional CarTooth cost

All other units are simulated on laptops or PDA so their cost is not included in the project's cost.

4 Summary

Our goal was to create a system for wireless payments for parking. To achieve this we have designed CarTooth, Parking Meter Unit and later the Administration Center. These parts make the core of our system and we have focused on developing them. They were thoroughly tested and are working now. Because of the flexibility of the designed core there is an opportunity to widen the range of possible usages. We have designed only a few from a variety of possible extensions. They are currently being developed. We have proposed efficient solutions to the problems occurring in wireless payments. For example we have coped with the problem: which device should be charged when more than one is in the range.

Making our system practically useful was very important for us. Since the very beginning we thought of it as a saleable product. We designed it to be inexpensive and have some additional facilities (like monitoring a car) to attract drivers. We have also created a plan which allows city authorities to put our system into practice smoothly. Our solutions permit not only the owner of the whole system to use Outside Units. Other companies can buy OUs and provide their services too. This will make our system grow faster and become even more popular.

It is worth mentioning, that we have changed disadvantages of Bluetooth into advantages. Its short range makes protection of cars better, because a car goes out of range almost immediately after being stolen, so alarm is fired shortly after the theft. Moreover, when a city has many Outside Units which connect often to AC, CarTooths can be localized with precision of 15 meters. It offers another potential usage of our system.

All protocols and units are made in such a way that in the future there is a possibility of smooth migration to a more advanced system. We have considered future development. Some kind of CarTooth could be integrated into a car. It would additionally use technologies like GPS and GSM (which are going to be built-in into modern cars). In that case constant connection with AC would be possible and online authorization of electronically signed transactions could be made

5 References

5.1 Literature References

[1] Ch. Petztold, Programming Windows, 5th edition, Microsoft Press, 1998.

[2] J. Nawrocki, B. Walter, A. Wojciechowski, Toward Maturity Model for eXtreme Programming, *Proceedings of 27th EUROMICRO Conference* (September 4-6, 2001, Warsaw, Poland), IEEE Computer Press, 233-239.

[3] K. Beck, *Extreme Programming Explained: Embrace Change*, Addison Wesley, 2000.

[4] W. Stalling, *Cryptography & Network Security: Principles & Practice*, 2nd edition, Prentice Hall, 1998.

[5] M. Kutyłowski, W. B. Strothmann, *Cryptography*, README, 1998 (in Polish).

[6] P. Metzger, *PC's Anatomy*, 6th edition, HELION, 2001 (in Polish).

[7] A. Rydzewski, *MCS-51 one chip microcomputer's family*, WNT, 1997 (in Polish).

[8] C. J. Date, *An Introduction to Database Systems,* WNT, 2000 (in Polish).

[9] Bluetooth specification, version 1.0B / 1.1, Bluetooth SIG, 1999 / 2001.

[10] ROK 101 008 Bluetooth Module, Ericsson Microelectronics, 2000.

[11] MSDN Library, Microsoft (Visual Studio .NET).

5.2 Internet References

[12] http://www.ericsson.com/

[13] http://msdn.microsoft.com/

[14] http://www.bluetooth.com/

[15] http://www.atmel.com/ (AT89C52 8-bit Microcontroller Datasheet)

[16] http://www.analogdevices.com/ (ADM232, ADXL202 Datasheets)