TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Automaatika instituut

ISS40LT

Dmitry Preobrazhenskiy 093945IASB

3D KRAANA JUHTIMISE MOBIILNE RAKENDUS IOS SÜSTEEMI JAOKS

Bakalaureusetöö

Eduard Petlenkov

Dotsent

Aleksei Tepljakov

Insener

Tallinn 2013

Autorideklaratsioon

Olen koostanud antud töö iseseisvalt. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud. Käesolevat tööd ei ole varem esitatud kaitsmisele kusagil mujal.

Kuupäev:

Autor: Allkiri:

Declaration

Herewith I declare that this thesis is based on my own work. All ideas, major views and data from different sources by other authors are used only with a reference to the source. The thesis has not been submitted for any degree or examination in any other university.

Date:

Author: Signature:

3D kraana juhtimise mobiilne rakendus iOS süsteemi jaoks

Annotatsioon

Selle töö põhieesmärk on luua mobiilne rakendus, mis võimaldab kontrollida 3D kraanaobjekti läbi Wi-Fi võrgu.

Seda rakendust tuleks kasutada siseruumides kraana läheduses ja see peaks töötama iOS seadmel iPad.

Rakendus peaks näitama ka informatsiooni objekti kohta, sealhulgas tõstevankri asendit, trossi asendit ja lasti nurka.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 25 leheküljel, 5 peatükki ja 8 joonist.

iOS mobile application for control of 3D crane

Abstract

The main goal of this work is to make a mobile application that allows controlling the 3D crane object through WI-FI network.

This application should be used indoors near the crane and should work on iOS device called iPad.

The application should also display the information of the object, which includes the position of the cart, the position of the thread, and the angle of the freight.

The thesis is in English and contains 25 pages of text, 5 chapters and 8 figures.

Table of contents

List of illustrations

1. Introduction

In modern world there is a need to create applications, that can display and possibly control remote objects. There are number of possibilities to use different devices for representation but some of them are better for certain tasks. For this purpose the iPad was chosen as an iOS device, because it has the parameters needed for this kind of work.

The main reason for that was that in fact, we are dealing with complex system, that consists of a regular PC, an Ethernet module that is plugged into the WIFI router, the control object itself, that is connected to the PC and MATLAB environment which serves the purpose of the interface, that links the hardware and software.

The iPad has the required WIFI module, which can be activated any time and connect itself with the system, making a closed info system. In this case, the iPad, is the client side that send/receives information, coding and decoding it. The PC and MATLAB serves as a server, which processes the information and the crane is the object that is being controlled.

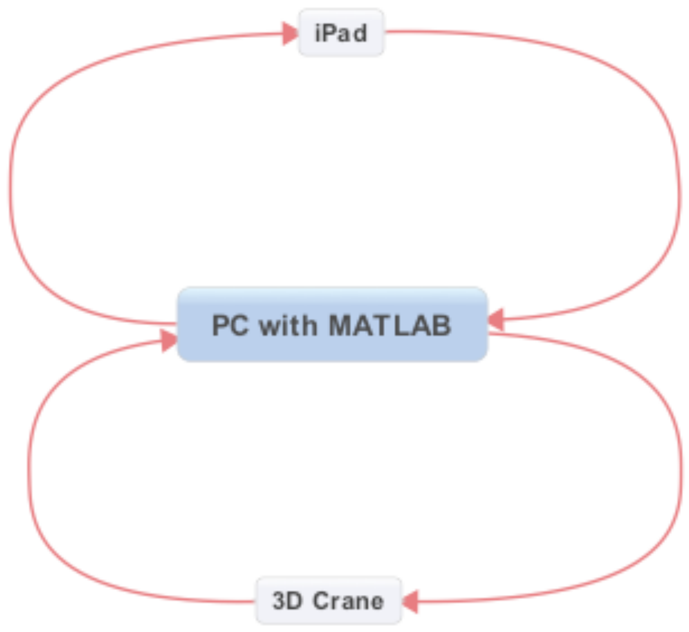


Figure 1. Dataflow

Based on the knowledge of the system, the mail goal of this work is to create an application that will be installable on the iPad and will be available in the App Store for download. This application can be used in different ways, but there are two main scenarios that are specifically required.

Firstly, the application should be used in the laboratory, to demonstrate how the control object works. This means, that the basic setup of the application should be understandable and should not require a lot of work. This also means that the interface should be user-friendly and display all the required information logically.

Secondly, the application should be used to check the physical specification of the control object, including sending specific values to the control object and resetting it's position to center and to home.

2. System description

2.1. 3D Crane object

The 3D crane is a non-linear electromechanical system having a complex dynamic behavior and creating challenging control problems. The system itself is controlled from a PC. The mechanical unit is supplied with a power supply, interface to a PC and a dedicated A/D, D/A board configured in Xilinx technology. [1]

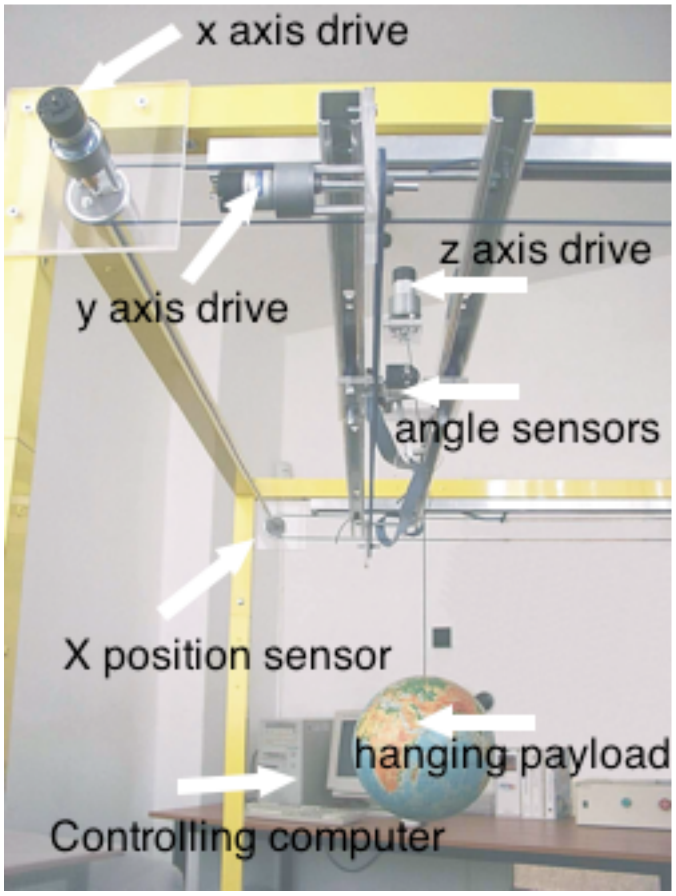


Figure 2. 3D Crane

The payload is lowered and lifted in the *z* direction. Both the rail and cart are capable of horizontal motion in the *x* direction. The cart is capable of horizontal motion along the rail in the *y* direction. Therefore the payload attached to the end of the lift-line can move freely in 3 dimensions and three DC motors drive the crane.

There are five identical measuring five state variables: the cart coordinates on the horizontal plane, the lift-line length, and two deviation angles of the payload. There are also 3 limit switches that do not allow the values to exceed their maximum to avoid damage.

The power interface amplifies the control signals, which are transmitted from the PC to the DC motors. It also converts the encoders pulse signals to the digital 16-bit form to be read by the PC.

The PC equipped with the RT-DAC/PCI multipurpose digital I/O board communicates with the power interface board. The while logic necessary to activate and read the encoder signals and to generate the appropriate sequence of the pulses of PWN to control the DC motors is configured in the Xilinx chip of the RT-DAC/PCI board. All functions of the board are accessed from the 3D Crane Toolbox, which operates directly in the MATLAB/Simulink environment.

2.2. MATLAB

MATLAB is a high-level language and interactive environment for different kind of tasks, such as numerical computation, visualization and programming. It can be used for different king of purpose, but what is most interesting, is to create a model for a control system. While the model itself can be programmed using the specific language, there is a specific tool that is used in conjunction with MATLAB, called Simulink. [2]

2.3. Simulink

Simulink is a block diagram environment for simulation and Model-Based Design, which provides a graphical editor, customizable block libraries, and solvers for modeling and simulating dynamic systems. [3]

Blocks, the main feature of Simulink, are mainly used to create a detailed block diagram of the system. They are highly customizable with functions that can be hand-written with MATLAB, C, Fortran or Ada.

There are many different block libraries available with standard Simulink package, but there are some specific tools required to create such a system, that is described in current work. These tools are used for rapid prototyping and HIL simulation, but what is mainly needed is the package called “Real-Time Windows Target”.

2.4. Real-Time Windows Target

Real-Time Windows Target provides a real-time engine for executing Simulink models on a Microsoft Windows PC and block that connect to a range of I/O boards, enabling to create and control a real-time system for rapid prototyping or hardware-in-the-loop simulation. [4]

The package comes with a library, which has blocks for input and output, varying from analog to digital. It also has two interesting sets of block, one of which is Packet Input – Packet Output and Stream Input – Stream Output. The main difference between them that the first one is used to send and receive unformatted binary data and the second one is used to receive and send formatted ASCII data. Both sets are operating with the “Standard Devices UDP Protocol”.

The Packet Input bock has a subset of parameters which can be configured/activated:

- Sample time
- Input packet size
- Block output data types
- Show “Data Ready” port
- Show “Data Error” port

Sample time – value that represents the frequency the block executes and interacts with the I/O hardware and synchronizes the model with the real-time clock at this sample rate.

Input packet size – number of bytes expected in each input packet. This number must be the same as the number of bytes required by “Block output data types”.

Block output data types – string or a cell array of strings that specifies how the data in each package obtained from the device is to be typed and grouped for input to the application. The block has an output port corresponding to each string. Each string has the format *[n*]datatype.* The data described by the string has the type specified by the datatype and the width specified by *n;* or 1 if *n* is not specified.

Show “Data Ready” port – indicates that the block has an output port that signals 1 if the block has new data available, and 0 otherwise.

Show “Data Error” port – indicates that has an output port that signals 1 if a data error has occurred, and 0 otherwise.

The Packet Output block has a same subset, with the difference that the Input is changed to Output. (Output packet size, Output packet field data types).

2.5. iPad

iPad is the line of tablet computers designed by Apple Inc, which carries onboard the Wi-Fi and cellular connectivity modules. The interface is build around device’s multi-touch screen, which also includes the virtual keyboard. The screen size is 197x148 mm and supports different orientations: portrait and landscape mode. [5]

The iPad only runs the software, which is either downloaded from Apple’s App Store or written by developers, who paid for a developer’s licence in registered devices.

In order to develop an application, a MacBook is needed with the developer’s studio called Xcode.

3. Modeling and Setup

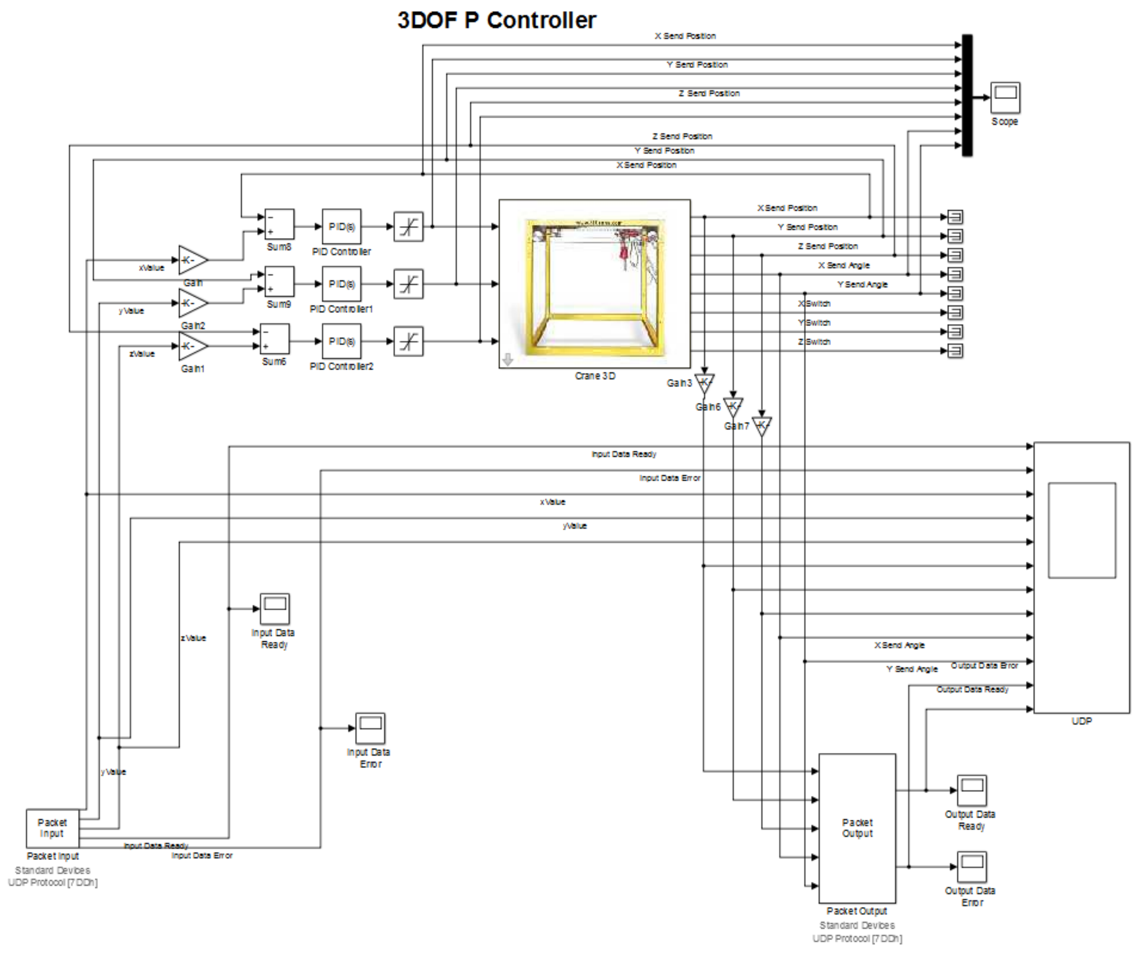
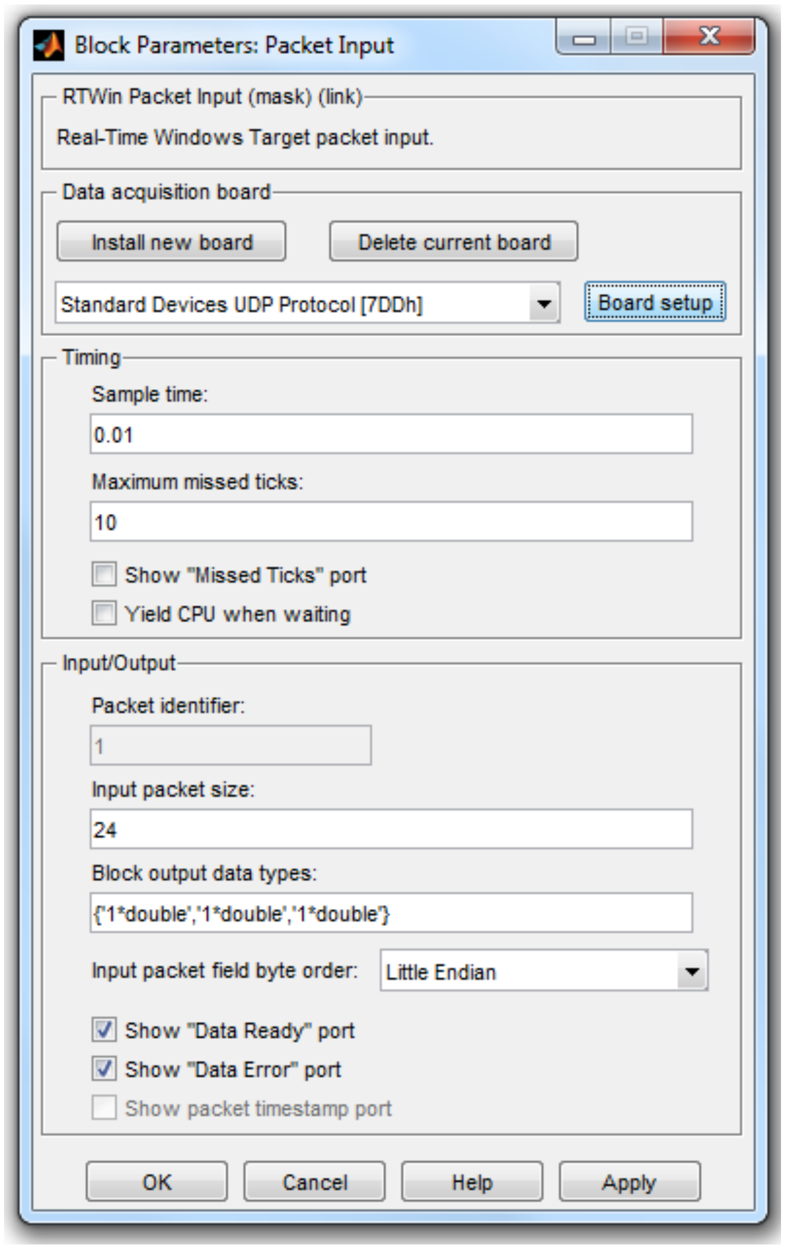


Figure 3. Simulink Model

3.1. Simulink Model

The template for this model was taken from a demo provided by Inteco LLC. The Packet Input and Packet Output blocks were added and configured with the correct settings. The xValue, yValue and zValues were amplified by using the gain blocks, which were configured using the Model Workspace constants.

3.2. Packet Input setup



RTWin Packet Input (mask) (link)

Real-Time Windows Target packet input.

Data acquisition board

Standard Devices UDP Protocol [7DDh]

Timing

Sample time: 0.01

Maximum missed ticks: 10

☐ Show "Missed Ticks" port

☐ Yield CPU when waiting

Input/Output

Packet identifier: 1

Input packet size: 24

Block output data types: {'1*double','1*double','1*double'}

Input packet field byte order: Little Endian

☑ Show "Data Ready" port

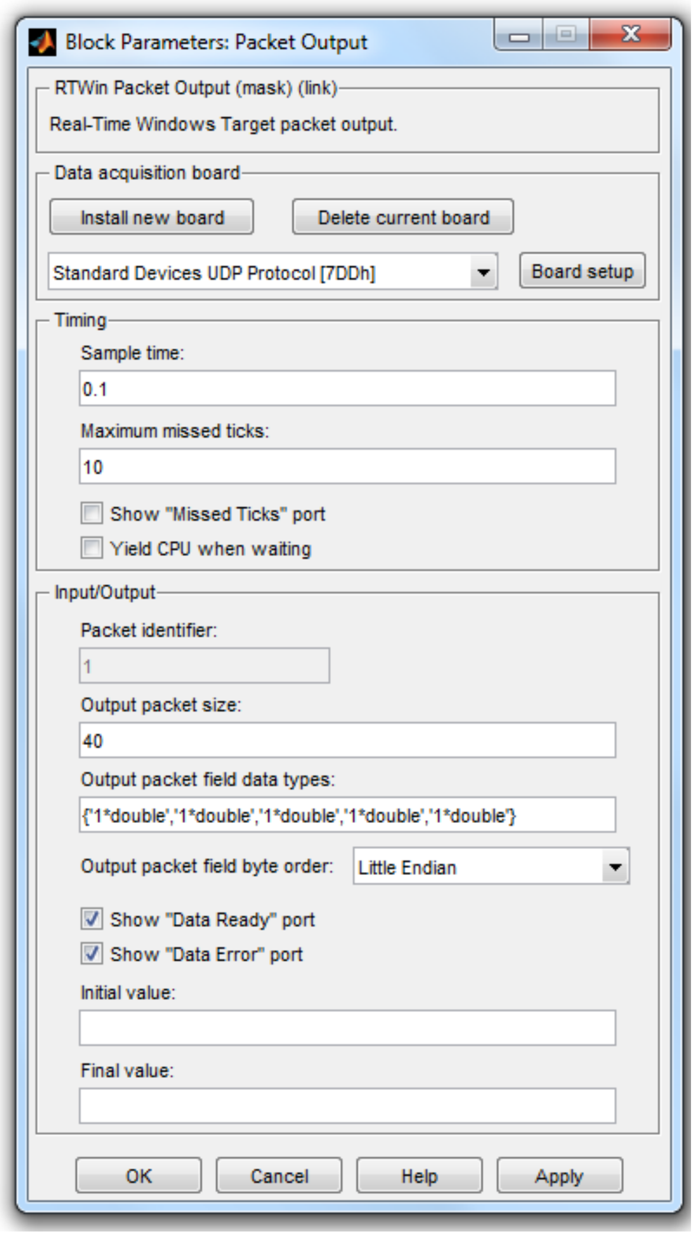
☑ Show "Data Error" port

☐ Show packet timestamp port

Figure 4. Packet Input

The main setup was to configure the Input packet size and Block output data types. It was decided that the iOS application would be sending a package with 3 double values, the new x, y, and z positions. Each double value is 8 bytes, so a total of 24 bytes were needed for the packet. Based on these requirements the *{"1*double", "1*double", "1*double"}* string was formed. [6]

3.3. Packet Output setup


Block Parameters: Packet Output

RTWin Packet Output (mask) (link)

Real-Time Windows Target packet output.

Data acquisition board

Standard Devices UDP Protocol [7DDh]

Timing

Sample time: 0.1

Maximum missed ticks: 10

☐ Show "Missed Ticks" port

☐ Yield CPU when waiting

Input/Output

Packet identifier: 1

Output packet size: 40

Output packet field data types: {'1*double','1*double','1*double','1*double','1*double'}

Output packet field byte order: Little Endian

☑ Show "Data Ready" port

☑ Show "Data Error" port

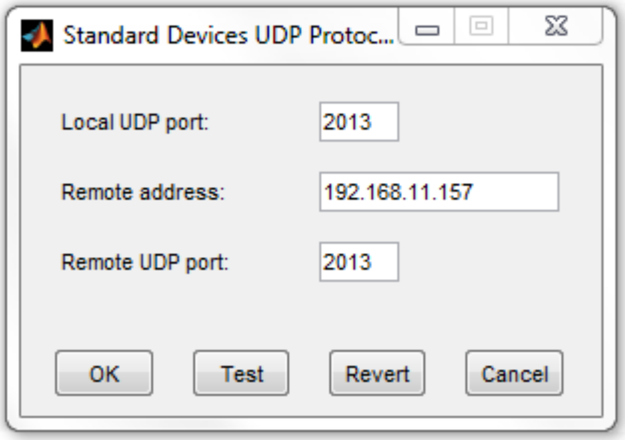
Initial value:

Final value:

Figure 5. Packet Output

The setup process for the Packet Output block is the same as Packet Input block, the difference is that instead of 3 double values, there are 2 more additional values needed. These are the two deviation angles of the payload. Based on these requirements *{"1*double","1*double","1*double","1*double" ,"1*double"}* string was formed. [7]

3.4. Shared Packet setup

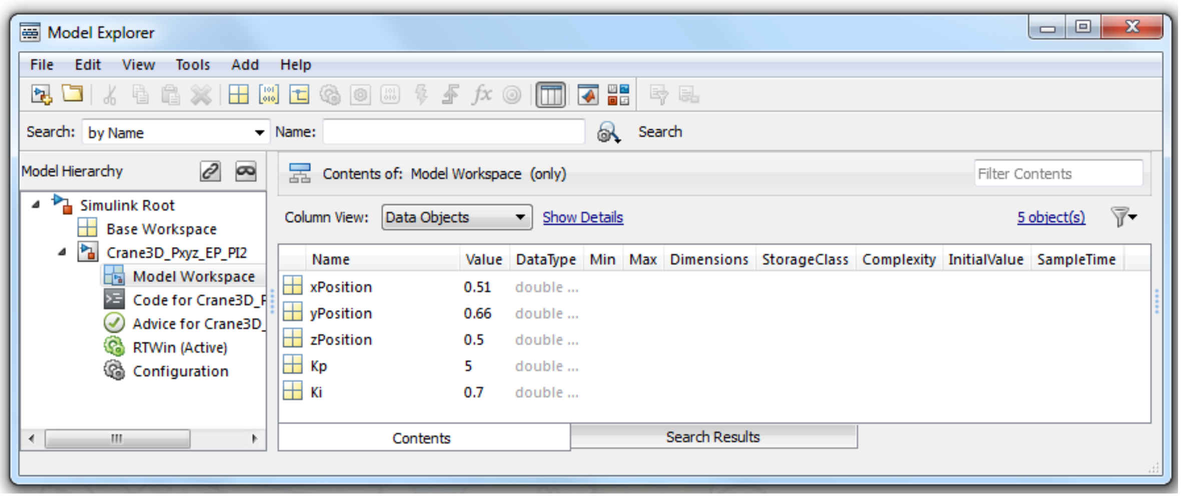

Standard Devices UDP Protoc...

Local UDP port:	2013
Remote address:	192.168.11.157
Remote UDP port:	2013

Figure 6. Shared Configuration

The Input and the Output Packet blocks both have the Board Setup Option, which openes this view. By standart, these blocks use 2013 UDP port, and the Remote address should the the client's address. The client in this work is the iPad with an application running on it.

3.5. Model Workspace setup



Contents of: Model Workspace (only)

Column View: Data Objects

Name	Value	DataType	Min	Max	Dimensions	StorageClass	Complexity	InitialValue	SampleTime
xPosition	0.51	double ...							
yPosition	0.66	double ...							
zPosition	0.5	double ...							
Kp	5	double ...							
Ki	0.7	double ...							

Figure 7. Model Workspace

The xPosition, yPosition are the maximum values of the cart and rail movements and zPosition the maximum lift-line length in meters. The Kp and Ki are PID controller's Proportional and Integral values, which were predefined.

3.6. Finalizing the setup

After all the settings have been configured and each time the client's address needs to be changed, the module should be rebuild using the *Ctrl+B* keyboard shortcuts or *Code* -> *C/C++ Code* -> *Build Model* in the upper toolbar.

After the build is done, the model should be connected to the target with the *Simulation* -> *Connect To Target* and the the model should be runned by *using Simulation* -> *Run* or the *Ctrl+T* keyboard shortcut.

If there are no error, the crane is set up and will begin receiving and sending UDP packets.

4. Application

4.1. Gathering information

UDP protocol is a simple transmission model with a minimum of protocol mechanism it does not have any guarantee of delivery, ordering or duplicate protection. One of its key features is that two computers can send and receive messages without prior communications to set up special transmissions channels or data paths. [8]

4.2. Requirements

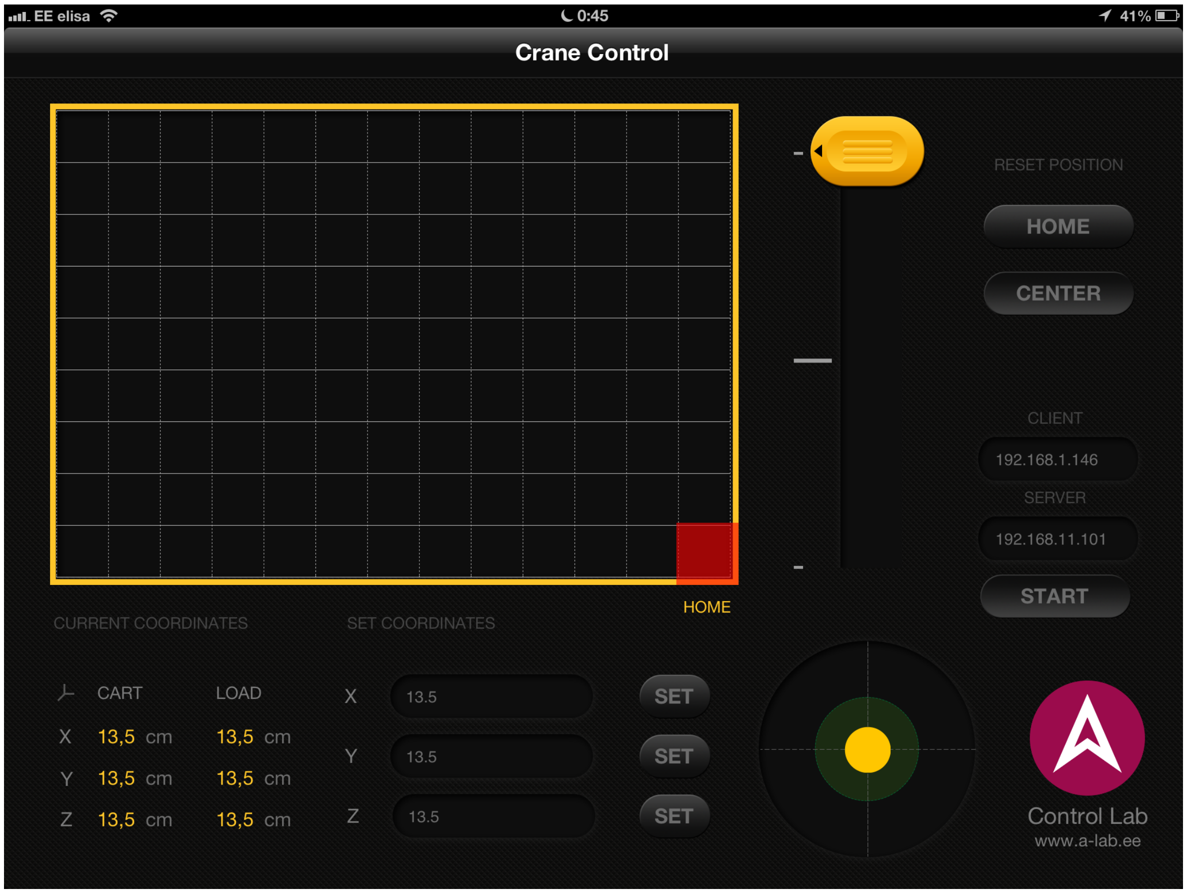
The interface of the application should consist of these parts:

- Grid representing the cart position
- Movable object that controls the cart
- Vertical slider that represents the position of the thread
- Movable object that controls the thread
- Circular zone representing the payload angles
- Visualization of current coordinates
- Inputs of new coordinates
- Two buttons to reset the position
- Input of server's IP
- Start/stop connection button

System requirements:

- Control should be done by user's gestures
- Application should accept manual input of the control values
- Application should be using Wi-Fi to connect
- Application should be installable to any iPad device
- Server's IP should be dynamically assigned

4.3. Interface realization



CLIENT 192.168.1.146

SERVER 192.168.11.101

	CART	LOAD
X	13,5 cm	13,5 cm
Y	13,5 cm	13,5 cm
Z	13,5 cm	13,5 cm

X 13.5
Y 13.5
Z 13.5

Figure 8. Application Interface

The colors of the application are chosen according to the specification and the design of the interface is build for landscape mode to fit all the required information and to have enough space to separate the information.

There are two zones that react to user's gestures, the grid with the yellow border and the vertical yellow slider. These two zones both react to standard tap and pan gestures. They are used to manually control the cart and the thread position. [9]

The manual input of the control values are done under the "Set Coordinates" section and are finilized after pressing the "Set" button. "Home" button resets the position of the cart and thread to their default values and the "Center" button moves the cart and thread to a predefined position.

4.4. Code realization

The are two main methods that convert values to UDP packet and parse incoming UDP package.

```
- (void)sendControlObject:(CCControlObject
*)controlObject {
    [controlObject
showAttributesWithBlock:^(NSDictionary *attributes,
NSError *error) {
        if (error) {
            [CCErrorHandler handleError:error];
            return;
        }

        double xPosition = [[attributes
objectForKey:xCurrentPositionKey] doubleValue];
        double yPosition = [[attributes
objectForKey:yCurrentPositionKey] doubleValue];
        double zPosition = [[attributes
objectForKey:zCurrentPositionKey] doubleValue];

        NSMutableData *sendingData = [NSMutableData
dataWithBytes:&yPosition length:sizeof(double)];
        [sendingData appendBytes:&xPosition
length:sizeof(double)];
        [sendingData appendBytes:&zPosition
length:sizeof(double)];

        [[CCSender instance]
sendDataWithBytes:sendingData];

    }];
}
```

```
- (void)retrieveObjectWithData:(NSData *)data {
    NSUInteger lengthOfData = [data length];
    NSUInteger sizeOfDouble = sizeof(double);
    NSUInteger numberOfObjects = lengthOfData /
sizeOfDouble;

    const void *bytes = [data bytes];
    double xReceivedPosition = 0;
    double yReceivedPosition = 0;
    double zReceivedPosition = 0;
    double xReceivedAngle = 0;
    double yReceivedAngle = 0;
    if (data.length && numberOfObjects == 5) {
        while (lengthOfData > 0) {
            memcpy(&yReceivedPosition, bytes,
sizeOfDouble);
            bytes +=sizeOfDouble;
            lengthOfData -=sizeOfDouble;
            memcpy(&xReceivedPosition, bytes,
sizeOfDouble);
            bytes +=sizeOfDouble;
            lengthOfData -=sizeOfDouble;
            memcpy(&zReceivedPosition, bytes,
sizeOfDouble);
            bytes +=sizeOfDouble;
            lengthOfData -=sizeOfDouble;
            memcpy(&xReceivedAngle, bytes,
sizeOfDouble);
            bytes +=sizeOfDouble;
            lengthOfData -=sizeOfDouble;
            memcpy(&yReceivedAngle, bytes,
sizeOfDouble);
            bytes +=sizeOfDouble;
            lengthOfData -=sizeOfDouble;
        }
    }
}
```

5. Conclusion

Writing this work started with gathering information and analyzing specific requirements for the upcoming application. Initially, the prototype was made for the iPhone, but due to the fact, that the interface has a lot of information, it was decided to keep only the iPad version.

Writing for iPad is always challenging, because this device supports different orientation and has a lot of space available to present the information. When making design, it should be taken in consideration how the end user will interact with the application, the main goal is to find the right balance between usability and specific requirements.

The 3D crane is not an ordinary object and building this type of applications from scratch should always start with first presenting a prototype before making and polishing the interface. The whole application has around 2000 lines of code including whitespaces but excluding the interface design.

The goals of the work were achieved and the application was send to the App Store for approval, which took around one week from review to the commence. By the time or writing, the application is avaiable world wide and can be download on any iPad with iOS 5.0+ [10]

The first problem encountered during writing the work was related to the fact, that the *X* and *Y* axis on the device are not the same as *X* and *Y* axis of the 3D Crane. This problem affected the way the gestures and the feedback was working on the device.

The second problem was related to the gestures, zones and how the objects are drawn on the screen. When using pan gesture, the movable object for the crane and thread should not leave its' zone and if we move the finger out of the zone, the gesture recognition should not end and correctly redisplay the movable objects.

The third problem was related to submitting the application for the review. Apple team has an unwritten rule, that when they are reviewing applications that are dealing with some real physicals objects, they need a demonstration video of how this application should work.

This application, if developed further, has a very big potential in terms of functionality and the tasks that it can handle. For example, the payload currently only displays its angles, but in a real life situation, this application would benefit, if it could calculate the route of the cart from the initial point to the destination, making such path, that the payload would be kept as stable as possible. Another benefit would be, it there could be additional visualization tools introduced in the application, such as web cam video that streams the 3D Crane movement. There is also a field of optimizations to be made in the Simulink model to make the movement more stable and more accurate.

The application code is also well structured and written using the paradigms of Object-Oriented Programming, meaning that the application can be adaptable to any object, that has the same physical properties.

This application will benefit anyone who is willing to learn MATLAB/Simulink models and who is interested in control systems.

List of used materials

[1] Inteco LLC, "3D Crane". [Online]. Available: http://www.inteco.com.pl/index.php?option=com_content&view=article&id=6&Itemid=12 [Accessed 05.06.13].

[2] The MathWorks Inc, "MATLAB". [Online]. Available: http://www.mathworks.se/products/matlab/ [Accessed 05.06.13].

[3] The MathWorks Inc, "Simulink". [Online]. Available: http://www.mathworks.se/products/simulink/ [Accessed 05.06.13].

[4] The MathWorks Inc, "Real-Time Windows Target". [Online]. Available: http://www.mathworks.se/products/rtwt/ [Accessed 05.06.13].

[5] Wikimedia Foundation Inc, "iPad". [Online]. Available: http://en.wikipedia.org/wiki/IPad#Screen_and_input [Accessed 05.06.13].

[6] The MathWorks Inc, "Packet Input". [Online]. Available: http://www.mathworks.se/help/rtwin/ref/packetinput.html [Accessed 05.06.13].

[7] The MathWorks Inc, "Packet Output". [Online]. Available: http://www.mathworks.se/help/rtwin/ref/packetoutput.html [Accessed 05.06.13].

[8] Wikimedia Foundation Inc, "User Datagram Protocol". [Online]. Available: https://en.wikipedia.org/wiki/User_Datagram_Protocol [Accessed 05.06.13].

[9] Wikimedia Foundation Inc, "Multi-touch". [Online]. Available: http://en.wikipedia.org/wiki/Multi-touch [Accessed 05.06.13].

[10] Apple Inc, "3D Crane Control". [Online]. Available: https://itunes.apple.com/ee/app/3d-crane-control/id656415163?mt=8 [Accessed 05.06.13].