KTH Electrical Engineering

Wireless Crane

Event-Triggered Control

AITOR HERNÁNDEZ, FAISAL ALTAF, JOSE ARAUJO

Stockholm July 19, 2011

TRITA-EE 2011:XXX
Version: 0.1

Contents

Chapter 1

Introduction

A Tower Crane [1] is used in the Automatic Control for academical purposes in some courses and it is a good case of study for Master Students and PhD, who can put their new control laws in practice.

The work and scenarios presented in this document has been developed by Faisal Altaf in his Master Thesis [4] and in the 19th Mediterranean Conference on Control and Automation [2].

Figure 1.1 shows the crane, with all the components involved to control it in a Event-Triggered scheme. The signals from the crane are splitted to be used in the PC using the FPGA provided by Inteco, and in the Event Detector (ED) which allows the microprocessor to detect the events.

To implement the Wireless Sensor Actuator Network we use the motes from TmoteSky [5], Telosb [10] or MAXFOR [3] using the TinyOS [11] operating system. For the wireless communications we used the IEEE 802.15.4 standard protocol [9] for low power communications implemented by TU Berlin, called TKN15.4 [6].

The code and software necessary to run the experiments explained below is available on the following URL:

`http://code.google.com/p/kth-wsn/source/browse/trunk/kth-wsn/apps.crane/`.

To compile the applications we recommend the use of the latest version of TinyOS that you could find on the following URL:

`http://code.google.com/p/tinyos-main/`

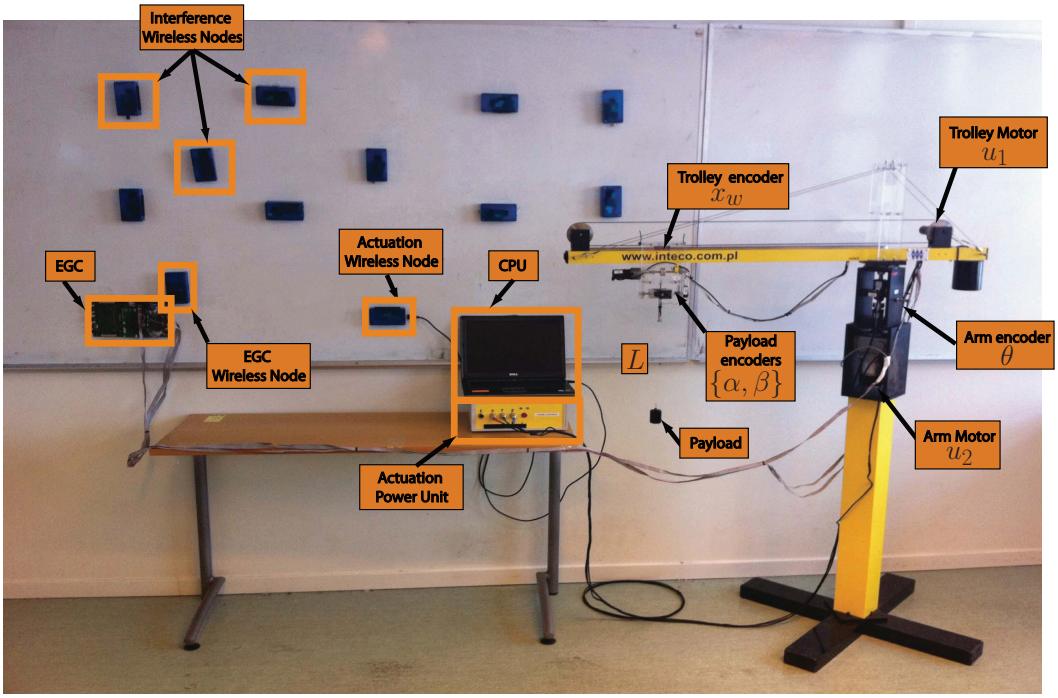


Figure 1.1: Experimental set-up for performing event-triggered control of tower crane lab process [1]. The system is composed of a tower crane, an Event-Triggered Generator and several nodes communicating over the IEEE 802.15.4 wireless network

Chapter 2

Requirements

Currently, this experiments can only be run in our lab, because it requires an specific hardware that has been designed by Faisal Faltaf at the Automatic Control department. The schematics and layout for the boards are not available, but the hardware is available.

Following we have the list with the different platforms and tools that we need to run the experiment. In the document [8], we explained that to work with the motes and TinyOS is extremely recommend to work in Linux, but here, we have an example where we use a Microsoft Windows enviroment.

- Tower Crane with its own power supply and FPGA to communicate with the computer. [1]
- ATmel STK600 development kit http://www.atmel.com/
- 2 motes. (Telosb, TmoteSky or MAXFOR)
- TinyOS properly installed [11]
- Source code for the motes http://code.google.com/p/kth-wsn/source/browse/trunk/kth-wsn/apps.crane/MotesFiles
- CygWin Shell with the Serial-forwarder from TinyOS in Windows, C version. [7] http://kth-wsn.googlecode.com/svn/trunk/kth-wsn/extra-tools/CygwinBin/
- LabView with the Vi which reads from the Serial-forwarder http://code.google.com/p/kth-wsn/source/browse/trunk/kth-wsn/apps.crane/LabViewFiles
- AVR Studio http://www.atmel.com/

- CodeVisionAVR http://www.hpinfotech.ro/

Chapter 3

Installation

First of all we need to connect the sensors cables from the crane to the board which sends the signals to the ED board and to the crane power supply. Then the ED board should be connected to the Event-Triggered Generator (EGC) board which includes the Atmel microprocessor. The last step is to compile the motes with the proper applications and connect them to the EGC and to the PC.

Figure 3.1 shows the boards used for this experiment. In this document we refer to the STK600 board as the EGC, and the PCB board with the encoders as ED.

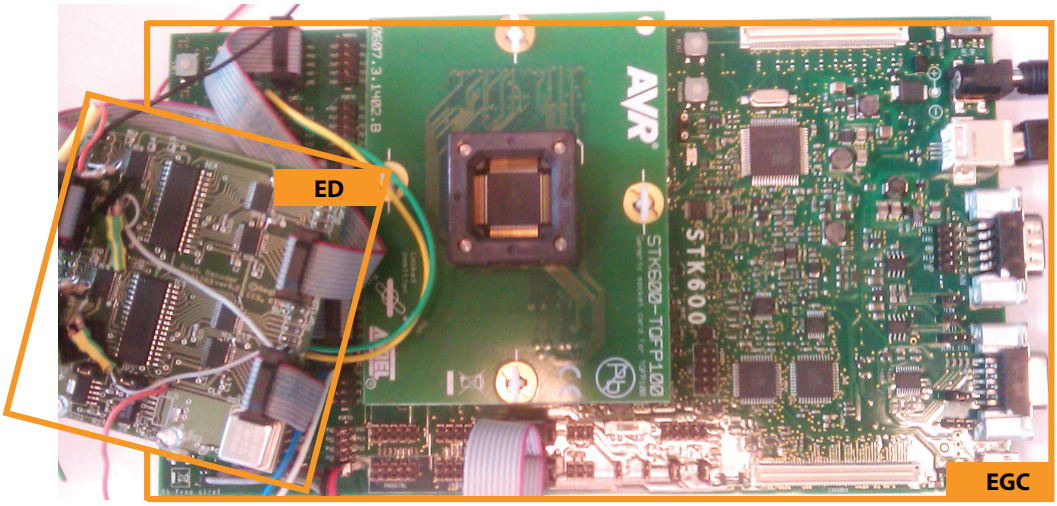
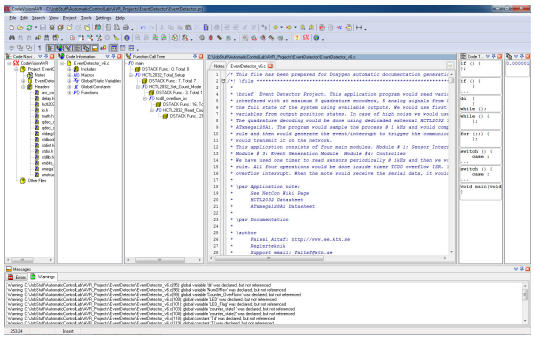
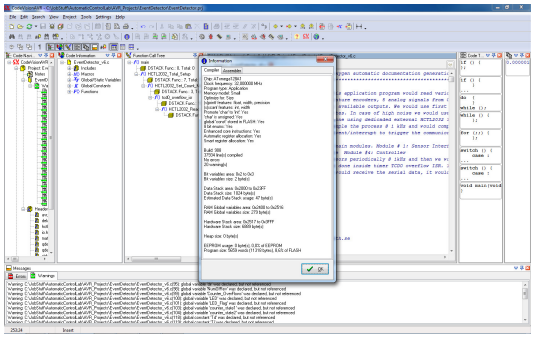


Figure 3.1: EGC with the ED connected



(a) Workspace



(b) End of compilation process

Figure 3.2: Screenshots of CodeVisionAVR

3.1 EGC board

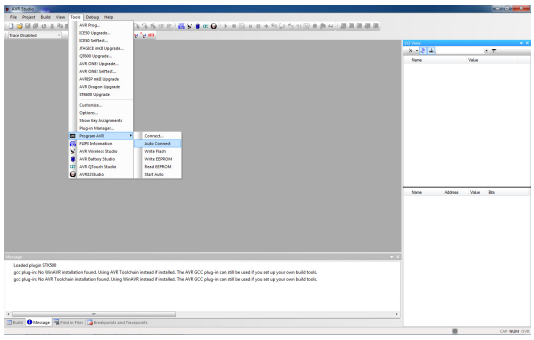
This board needs to be connected with the adaptor provided with the development kit. When the power cable is connected and the power switch is on, the green led should be on. Regarding the connections would be simpler to show the figures of the current system:

http://code.google.com/p/kth-wsn/source/browse/trunk/kth-wsn/apps.crane/BoardFigures

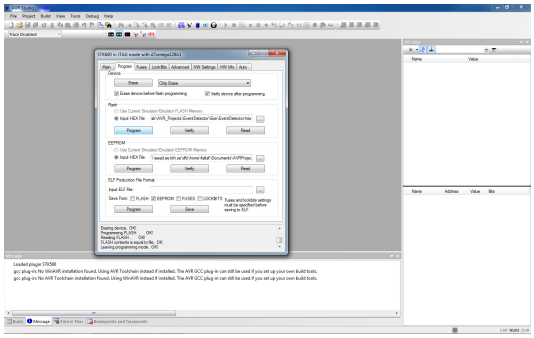
We will describe the steps to program the microprocessor with the `EventDetector` program. First we start generating the project with the *CodeVision AVR*, and once we have the application compiled, we flash the application to the microprocessor with the *AVR Studio*.

Compilation

1. Start the *CodeVision AVR*.



(a) Connection to the board



(b) Program tab

Figure 3.3: Screenshots of AVR Studio

2. Open the project `EventDetector` that could be found in the `uCFiles/` folder. Figure 3.2a shows the workspace with the `EventDetector` project opened.

3. Press Compile and Build buttons or press Build all to create the .hex file. Figure 3.2b shows the pop-up dialog that appears when the building process has finished succesfully.

Program

1. Start the *AVR Studio*

2. Go to the Tools menu -> Program AVR -> Auto connect. Figure 3.3b shows where is the Auto-connect item.

3. In the *Main* tab, be sure to have the following configuration:

   - Select Device (MCU): ATxMega128A1
   - Programming mode & Target Settings: JTAG
   - To check if the board is connected, press “Read signature”

4. In the *Hardware settings* tab:
   - Set to 3.3V
5. In the *Program* tab. (Figure 3.3b).
   - Program mode: Flash
   - Input Hex file: `EventDetector/Exe/EventDetector.hex`
   - Press "Program" button. The orange LED of the board should blink during the flashing process.

3.2 ED board

This board needs to be supplied by an external power supply. It allows a voltage levels between $9 < V_{cc} < 10V$. Check pictures for the configuration.

3.3 Motes

For this experiment we need two motes: one is connected to the EGC and another to the computer. It is important that the mote is properly connected to the EGC. Figure **??** shows the connection. The white cable is connected to the PIN 2 of the expansion header and the blue cable is not connected. Moreover, the grey cable is connected to the GND, which is the PIN number 9, and the brown cable is not connected.

3.3.1 EDReceiver

The mote connected to the EGC has the `EDReceiver` application that can be found in the folder `MotesFiles/EDReceiver`. To program the application:

```
make tmote install bsl,/dev/ttyUSBXX
```

where XX is the port number. Mention that we do not need to set manually the ID because is fixed to 0x10 in the header file `app_crane.h`

3.3.2 Actuator

The mote connected to the PC has the `Actuator` application that can be found in the folder `MotesFiles/ActuatorPCNoBuffer`. To program the application:

```
make tmote install bsl,/dev/ttyUSBXX
```

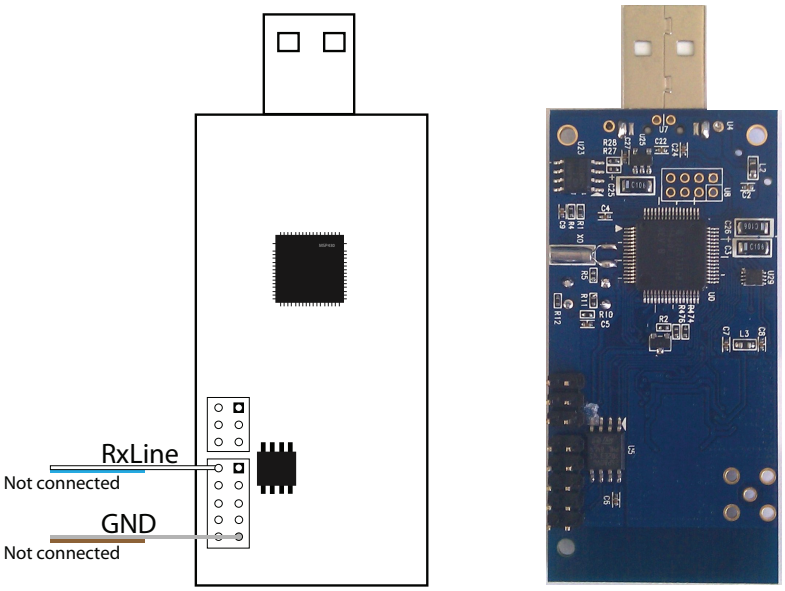


Figure 3.4: Connection between the EGC and the mote

where XX is the port number. Mention that we do not need to set manually the ID because is fixed to 0x20 in the header file `app_crane.h`

Although it is optional we can have an sniffer mote, that provides information of the communication performance. The application is called `Sniffer` and it could be found in the folder `MotesFiles/Sniffer`.

3.4 LabView

LabView does not require extra configuration. To run the file you only need to open the `Controller_TCrane_MoreData.vi` placed in the `LabViewFiles/` folder.

Figure 3.5 is an screenshot of the LabView file. It shows the *Main* tab, where we have some visualization charts, button to turn on/off the motors and a pop-up menu to change the controller and references. The references are only valid for the P (tracking) controller.

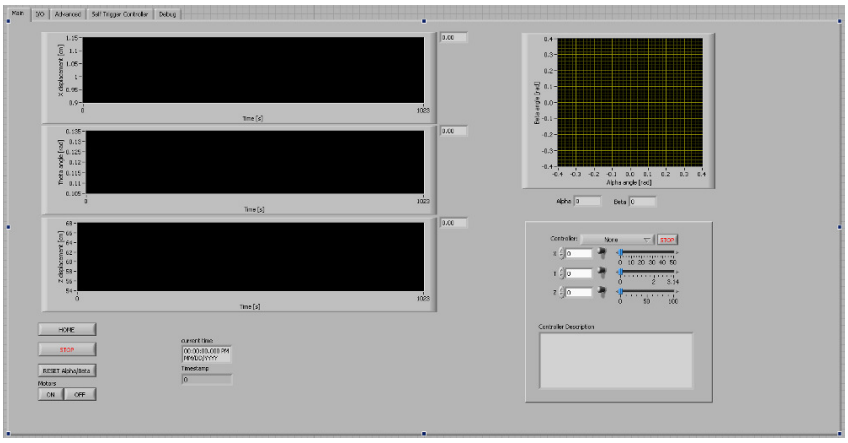


Figure 3.5: Screenshot of the LabView file to control the Tower Crane

Chapter 4

Usage

In the previous chapter we learned how to install all the necessary components to run the experiment. At this point, we show the steps that are needed to run it, including the use of the Serial-Forwarder, LabView and EGC.

Figure 4.1 shows a simple diagram with the steps to run a certain controller using the EGC.

Serial-Forwarder As we have seen in [7] the Serial-Forwarder is the application that acts as a bridge between the serial port and the PC, creating a TCP/IP server. What we are going to do here is to create this server, and connect to it form LabView to read the packets.

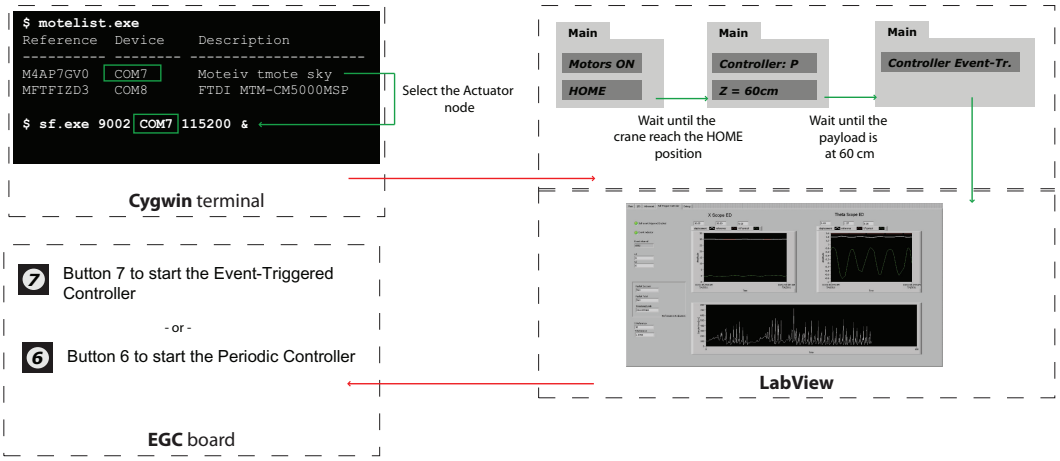


Figure 4.1: EGC board with the important buttons

1. Start CygWin Shell
2. Run the motelist to get the port number of the mote. `$ motelist.exe`
3. Get the port number of the Actuator mote, the mote that is connected to the computer
4. Start the Serial-Forwarder in the previous port. `$ sf.exe 9002 COMX 115200 &`
5. If we want we could open a listener, to check if the connection is correct. `$ sflisten localhost 9002`. If the ED is sending packets, we should see them now.

LabView With the Serial-Forwarder running in the background, it is time to start the LabView. The LabView files and dependences could be found in the folder `LabViewFiles/`. To start open the file called `Controller_TCrane_MoreData.vi` and follow these instructions:

1. Run the Vi and let it run to check if any error appear. Check the *Advance* tab if there if any error with the FPGA. Check the RT-DAC/USB status panel. If there is an error check:
   - Is the FPGA connected? If it is, restart LabView if not ....
2. Set the crane in the initial position. Go the *Main* tab, enable the motors and press the *Home* button.
3. Wait until the crane is at the "home" position.
4. Start the P controller and set the payload reference to 60 cm ($Z = 60cm$).
5. Wait until the payload is set at 60 cm.
6. Select the Event-Triggered Controller. Now, we can control the crane with the EGC board.

EGC Figure 4.2 show the EGC board (ATmel STK600) with the buttons or switches. It describes their programmed functionality.

At this point, LabView is waiting for message from the mote. It will not receive any packet until we start the algorithm in the EGC. To do it, follow these steps:

1. Press the "reset" button, to clear all the previous data stored.
2. Press the button "7" to start the Event-Triggered controller or press the button "6" to start the Periodic Controller

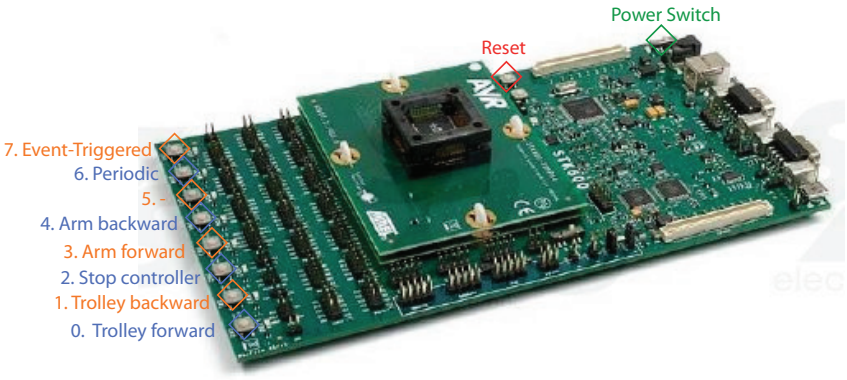


Figure 4.2: EGC board with the important buttons

To stop the EGC press button “2”.

Chapter 5

Results

In this chapter we show different ways of logging and showing the data. We also use LabView to log the data, because the FPGA allows us to get the signals from the crane every 20 ms.

On the other hand, we use a modified version of the `seriallisten` to log extra information. With this program running over a sniffer node, `Sniffer` application, we can analyse the communication performance and time stamping the packets with the exact timing where the mote has received a packet.

5.1 Logging

LabView is configured to store the data in a file automatically. By selecting the "Self-Triggered Controller", it automatically save a file with all the data from the sensors and actuators values. By default the file is saved in the folder `matlab/output`. The file created is already a `.m` file and it can be loaded directly in Matlab to import the data.

On the other hand, if we want to have accurate values for the inter-event time and analyse the communication performance we need to run the `seriallisten` and store the output in a file. For this, use the following command (in a Linux machine)

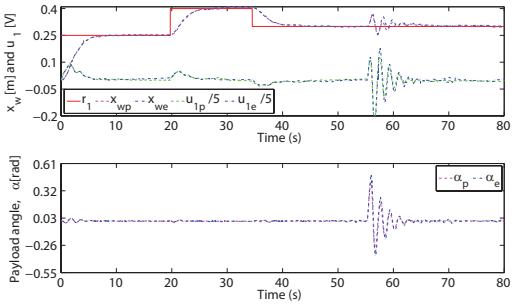
```
$ ./seriallisten /dev/ttyUSB0 115200 >
/home/kthwsn/workspace/apps.crane/matlab/sf_output/data_20110704_1658
```

5.2 Plotting

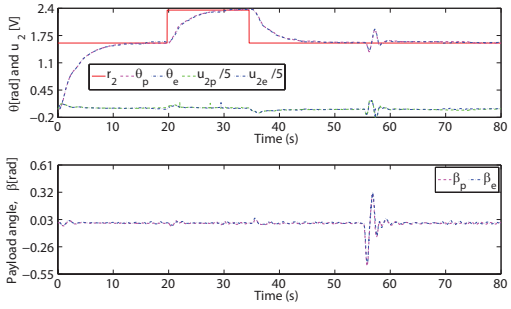
After the experiment is done, and all the data is gathered is time to show the results and plot the figures. For this purpose, we have a Matlab function that, by giving the two files (LabView and seriallisten) for the Self-Triggered and Periodic experiments, shows all the results. For example, by running:

```
> Plots_TT_ET('outputs/data_201102102230';'sf_output/tt_201102102230',
'outputs/data_201102102226'; 'sf_output/et_201102102226')
```

we get the results and comparative between two experiments without any interferences with Self-Triggered and Periodic controllers. Figures **??** show some examples.



(a) Time response of Trolley subsystem and control input to the trolley motor.



(b) Time response of Arm subsystem and control input to the arm motor.

Figure 5.1: Comparison between a periodic time-triggered controller (TTC) and an event-triggered controller (ETC) for step reference tracking with no losses.

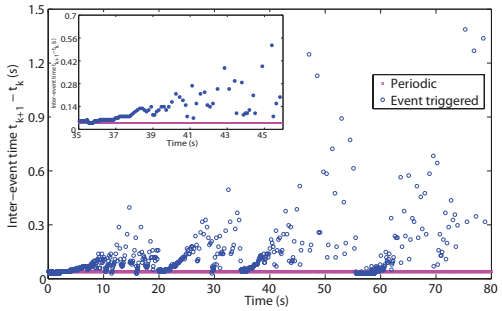


Figure 5.2: Inter-event times $t_{k+1} - t_k$ for a periodic time-triggered controller (TTC) and an event-triggered controller (ETC) for step reference tracking with no packet losses ($N_L = 0$) and no external nodes ($M_{nodes} = 0$)

References

[1] Inteco. tower crane. Technical report. URL http://www.inteco.com.pl/.

[2] *Wireless Event-Triggered Controller for a 3D Tower Crane Lab Process*. 19th Mediterranean Conference on Control and Automation, jun. 2011.

[3] Advantic. Maxfor data sheet. Technical report, Madrid, 2010. URL http://www.advanticsys.com/files/CM5000.pdf.

[4] Faisal Altaf. Modeling and event-triggered control of multiple 3d tower cranes over WSNs. Master's thesis, Royal Institute of Technology (KTH), November 2010.

[5] Moteiv Corporation. Tmote sky data sheet. Technical report, San Francisco, June 2006. URL http://www.bandwavetech.com/download/tmote-sky-datasheet.pdf.

[6] Jan-Hinrich Hauer and Adam Wolisz. TKN15.4: An IEEE 802.15.4 MAC Implementation for TinyOS 2. Technical report, Technical University Berlin - Telecommunication Networks Group, March 2009. URL http://www.tkn.tu-berlin.de/publications/papers/TKN154.pdf.

[7] Aitor Hernandez. Communication between pc and motes in tinyos. Technical report, Royal Institute of Technology (KTH), July 2011.

[8] Aitor Hernandez. Getting started with tinyos at the automatic control lab. Technical report, Royal Institute of Technology (KTH), July 2011.

[9] IEEE Std 802.15.4, Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs) , September 2006. URL http://standards.ieee.org/getieee802/download/802.15.4-2006.pdf.

[10] Crossbow Technology. Crossbow telosb. Technical report, San Jose, California. URL http://www.willow.co.uk/TelosB_Datasheet.pdf.

[11] TinyOS. Tinyos. Technical report, 2010. URL http://www.tinyos.net/.