WIRELESS PERFORMANCE OF AN EMBEDDED VIDEO ACQUISITION SYSTEM

RAKUN, J.; BERK, P.; OCEPEK, M. & LAKOTA, M.

Abstract: *In this work, we present and asses wireless performance of a data acquisition system developed to capture video stream on a field robot. The system runs on ARM Cortex A8-based embedded computer and uses The imaging source's DBK31BU03 digital camera that is able to capture 30 fps at a resolution of up to 1024 × 768 pixels. The role of the acquisition system is to receive a data request from the client and respond with (send) the last valid video frame that was captured from the digital camera. The acquisition system uses TCP link so the requesting client can be another program on the embedded computer or a program located on a different workstation connected via wired or wireless Ethernet link. The results show that the embedded system is fast enough to transmit on average 3 and up to 10 (partial) frames per second using a wireless IEEE 802.11g connection.*

Key words: *embedded computer, data acquisition, field robot*

Authors´ data: Dr. Sc. **Rakun**, J[urij]; Dipl.-Ing. **Berk**, P[eter]; M. sc. **Ocepek**, M[arko]; Univ. Prof. **Lakota**, M[iran], Faculty of Agriculture and Life Scineces, University of Maribor, Pivola 10, 2311 Hoče, Slovenia, jurij.rakun@uni-mb.si

This Publication has to be referred as: Rakun, J[urij]; Berk, P[eter]; Ocepek, M[arko] & Lakota, M[iran] (2011). Wireless Performance of an Embedded Video Acquisition System, Chapter 34 in DAAAM International Scientific Book 2011, pp. 425-432, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-901509-84-1, ISSN 1726-9687, Vienna, Austria
DOI: 10.2507/daaam.scibook.2011.34

1. Introduction

Automating everyday tasks that farmers or food producers have to do seems very appealing but in reality presents quite a challenge. While large food producers rely on the use of heavy machinery to automate frequent and repeating tasks, this is still not the case for mid- and small-sized farms, which can pose a potential food safety problem. If handled manually, food can transmit disease from person to person as well as serve as a growth medium for potentially harmful bacteria (Beuchat, 2002). Nevertheless, some work still demands manual labour that is time consuming, exhausting and expensive. The introduction of a small army of intelligent robots to do the job quicker and more accurately is appealing but is not possible yet. For instance, the natural uncontrolled environment poses a challenge with its changing conditions. An overview on the subject (Stajnko et al., 2009) showed that there are some potentially good solutions but the authors rely on specific conditions (like night time) or their solution is designed to work in controlled environments (green house) and some are simply too big or too heavy to be useful at this stage. In this paper, the problem is tackled by introducing a mobile agricultural platform.

In order to achieve the goal, we a small autonomous self oriented robot was built that could for instance serve as a potential tool for selective pesticide spraying (Thompson et al., 1991), fertilizer applicator (Nobutaka, 1990), fruit picking robot (Bulanon et al., 2001) or even as a device that could estimate the yield at the end of the harvest by simply taking digitised images of the tree canopies (Stajnko et al., 2009).

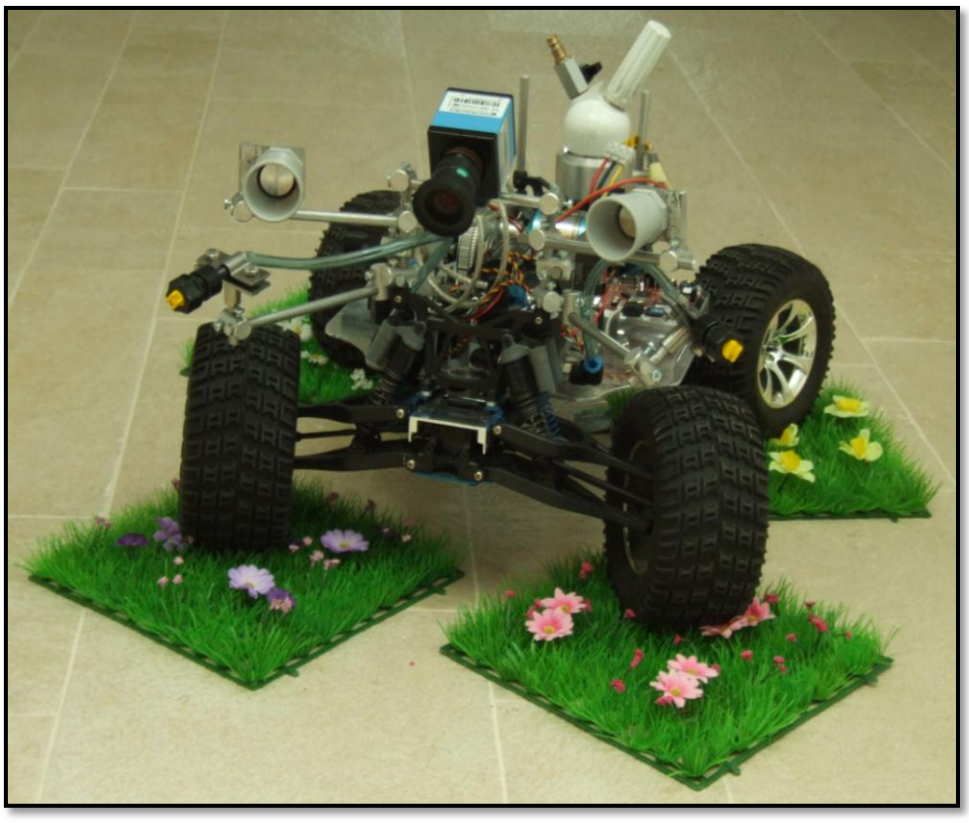

Fig. 1. Image of the Field Robot

2. Materials and methods

We solved the first step and build a hardware platform, but we still face more challenges in developing a good software platform. This is the reason the algorithms are currently developed and tested on a workstation, where live images from the robot are used. As we are working with wireless link to transmit each image frame it needs to be analyzed and evaluated, which is the purpose of this paper.

The field robot consists of four critical components. The first is an onboard embedded computer connected to a high speed digital camera and a wireless connection. The second is a four-wheel drive that enables the platform to move around the rough terrain. The third component of the robot is the set of ultrasonic sensors that helps to keep track of the surroundings. And finally, the fourth part, the onboard reservoir and nozzles that spray the plants. The robot layout is explained in detail in Fig. 2.

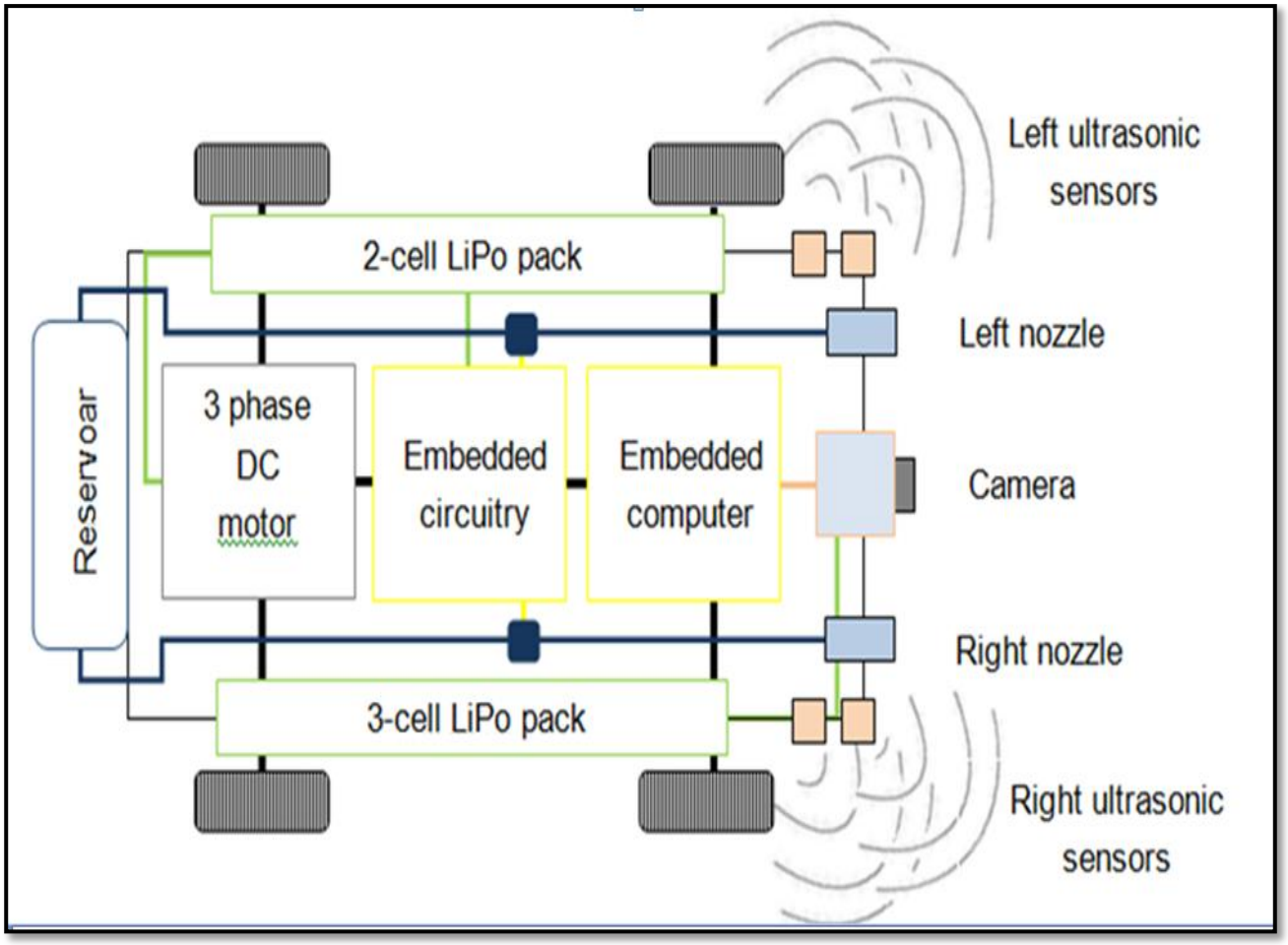

Fig. 2. The Field Robot – component layout

The support circuitry of the robot is built around an 8-bit AVR microcontroller (Atmega324p) that measures the distance from the sensors to the closest obstacle and navigates the robot according to the requests it receives on the USART port. The port is currently connected to an XBee pro wireless module (Digi International Inc.), as the navigation program for the time being runs on an off-field development

workstation, and will be connected directly to the onboard embedded computer later in the development stage.

The navigation logic of the robot relies on the distance measurements captured by using ultrasonic sensors. We are using Maxbotix WR1 ultrasonic sensors (Maxboticx Inc.) that are waterproof, and therefore suitable for outdoor conditions, but unfortunately have a narrow detection cone. This is the reason why four sensors were used and placed in the front of the robot as our intent was to reduce the chance of hitting an obstacle.

In order to drive the mobile platform, a high-performance brushless motor X-power eco A4130-06BL was chosen that is controlled by an X-power professional 70-3P BEC controller. It is able to automatically detect the cell number and type of batteries used in the setting. In our case we are using Lithium-polymer batteries (LiPolice 5000mAh/2S 7,4V 75/100A) with two cells. The controllers' main job is to control the speed and direction of the motor.

2.1 An embedded computer

The onboard embedded image acquisition part consists of three main components; a high resolution digital camera that captures video stream, a wireless interface (Asus WL-167g) that works according to the IEEE 802.11g standard and an embedded computer for which the Beagle Board ver. C4 was selected. This is a superscalar ARM Cortex A8-based embedded computer that consumes only around 2 W of power and gives 1200 Dhrystone MIPS performance.

For the embedded computer operating system, we have chosen and customized a version of the Linux operating system (Matthew & Stones, 2004) - an Angstrom distribution. The first step we took was to install only the necessary software so the final version fitted on a 1GB SD memory card. As the second step, we compiled a custom 2.6.32 kernel according to the hardware specifications and finally, as the third step, we tuned the settings for the TCP/UDP protocols (TCP tuning guide, 2010) to reserve more buffer space and to transmit small data chunks faster in effect minimizing the transmission delays. We have increased the write/read TCP buffer sizes to 16 MB, TCP buffer limits auto-tuning to a minimum of 4KB and a maximum of 16 MB of memory. In addition the size of the TX queue length also had to be increased and was set to 5000 bytes. All suggested modifications helped to minimize the transmission delays that will be summarized in the results section.

2.2 Server algorithm

The main role of the custom-written server program running on an embedded computer is to provide the off-field computer with video stream. It is able to capture live stream at the rate of 30 frames per second with resolution of up to 1024 x 768 pixels and send it to an off-field workstation in Bayer encoded format (Gonzales & Woods, 2008) using different resolutions. The server program is also able to provide the client with partial images, if we only want to observe part of an image and whole image frames are not needed.

In order to be able to capture video stream and to transmit the last captured image simultaneously two memory locations are reserved (Matthew & Stones, 2004). The algorithm first chooses one of the locations and it stores a new image. It then continues using the second memory location and stores another, newer image. This is repeated until the off-field workstation requests a new image. In this case, the location of the last fully saved image is protected so it is not overwritten by the video stream capturing thread unit it is not fully transmitted. The video capture thread stores new images in one of two locations that is available for writing at the time when it is not locked by the server thread. In this way, we provide up-to-date data and also prevent the transmission of incomplete images. The mechanism for capturing and transmitting images is summarized in Fig. 4.

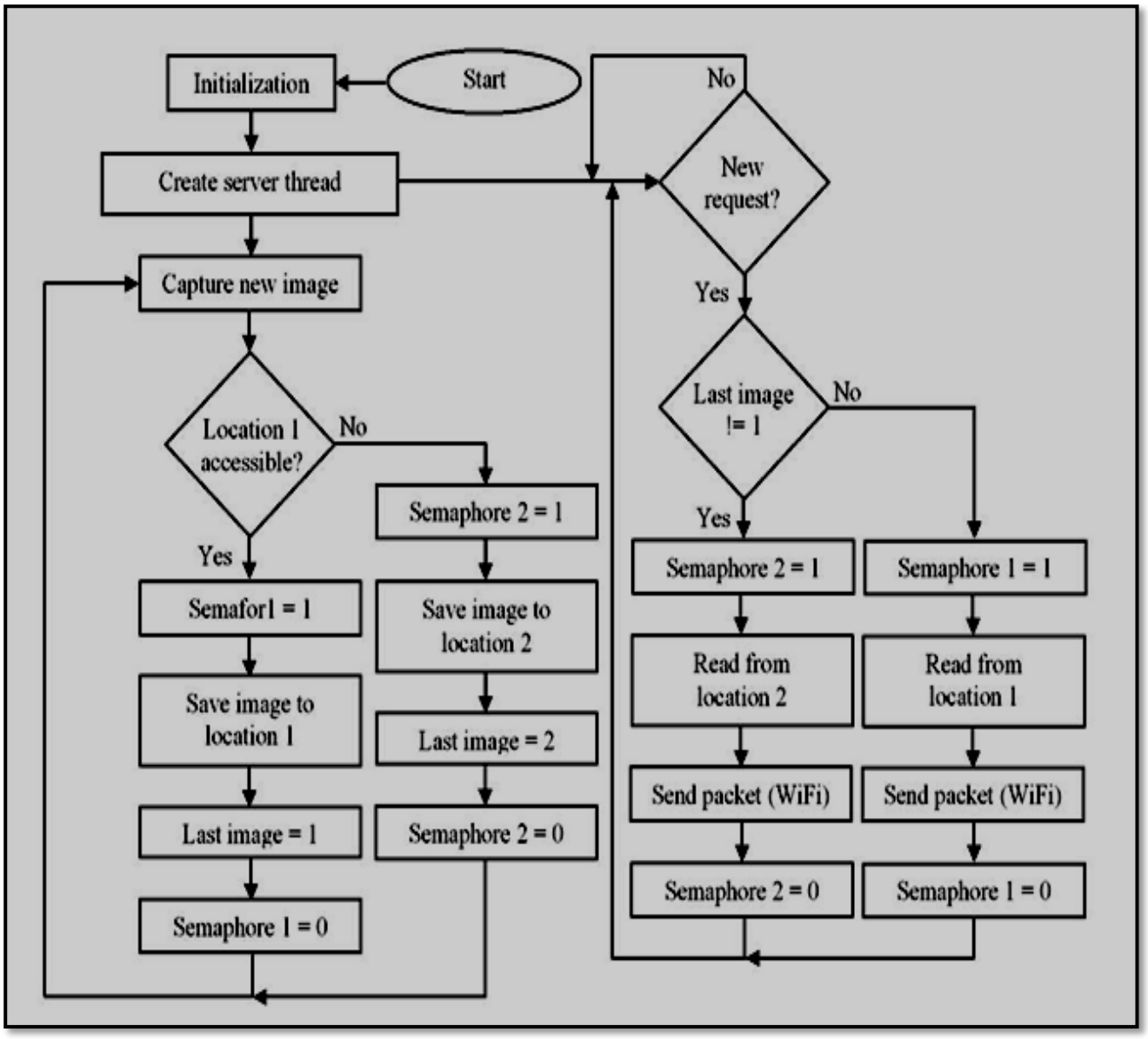

Fig. 4. Flow chart of the server algorithm.

The server program was written in C and compiled with default optimizations using GNU Complier collection (GCC) ver. 4.3.3.

3. Results

It should be stressed that partial frames (one third original size or 1024 × 262 pixels) were transmitted as we are currently only processing the lower part of an image representing the soil and smaller weed plants. The request package payload was 3 bytes (+ TCP overhead) and the response payload was 268 kB long (+ TCP overhead).

We recorded time measurements using 10 different iterations of image transfers where within each iteration 20 requests were sent and 20 images were received. Time was measured for each transfer from a point when the request was sent and the whole image frame was received. Average, minimum and maximum times were recorded for each of the iterations as shown in Table 1.

Iteration	Maximum time [s]	Minimum time [s]	Average [s]
1st	1.15	0.10	0.36
2nd	1.05	0.09	0.34
3rd	1.15	0.11	0.40
4th	0.99	0.08	0.34
5th	1.01	0.07	0.31
6th	0.98	0.10	0.38
7th	0.99	0.09	0.40
8th	0.95	0.07	0.39
9th	1.00	0.08	0.35
10th	0.96	0.08	0.33
Average [s]	1.02	0.09	0.36

Tab. 1. Ten iterations of data transmission; within each iteration, 20 frames were requested/received and maximum, minimum as well as average transmission times were recorded

4. Discussion

On average, it took 0.36 seconds to transmit each frame while, in the worst case scenario, the delay can reach 1 second and, in the best case, just 0.09 seconds. Further analysis showed that the maximum times are always recorded at the beginning of each transfer session and once the transmission settles, it takes, on average, 0.09 seconds to transfer an image. Based on the measurements, we conclude that it is advisable to start the transmission of the images at least 1.5 second before they are actually used and analysed on the workstation. The other approach would be, to start the transfer of an image data a few seconds before they are actually needed, and discard the unneeded data at the receiver's end, just to settle the transfer speed of the wireless stream.

5. Conclusions

The robot presented is still in the development phase but already offers unique possibilities. The basic outline is set; the robot can navigate, communicate with the remote workstation, accepts and carries out commands when to open and when to close the nozzles.

In this work, we summarized the server acquisition program. As we are still in the development phase, we heavily rely on the use of wireless connection. This is the reason we tested the speed of transmission, in order to inspect if the delay might pose a problem. As each partial image frame depicts 0.5 m of scene in front of the robot and as the robot currently moves at maximum 1 m/s, we can safely conclude it does not pose a problem as long as we include 1.5 s time before each test, as we described in the previous subsection. The robot sends images that depict 0.5 m of scene before the robot which means that the frames will overlap for at least 0.4 m (the current top speed of the robot is 1 m/s).

One of the crucial improvements for future work is to include JPEG compression. The data needed to be sent will be minimized which we expect will outweigh the time needed to compress and decompress each frame. We also plan to replace the IEEE 802.11g network equipment with the one that supports IEEE 802.11n, which theoretically would increase the performance by a factor of 3 at 150 MB/s or 6 at 300 MB/s.

6. Acknowledgements

We would like to thank The imaging source GmbH for providing us with a digital camera we used in this research work.

7. References

Beuchat, L. R. (2002). Ecological factors influencing survival and growth of human pathogens on raw fruitsnext term and vegetables, *Microbes and Infection*, vol. 4, pp. 413-423

Bulanon, D. M.; T. Kataoka; Y. Ota & T. Hiroma (2001). A Machine Vision System for the Apple Harvesting Robot, *Agricultural Engineering International: the CIGR Journal of Scientific Research and Development*, vol. III, pp. 1-11

http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/zigbee-mesh-module/xbee-zb-module.jsp#overview - Digi International Inc., *Accesed on: 2010-10-20*

Gonzales, R. C. & Woods, R. E. (2008), Digital Image Processing, 3rd edition, Prentice Hall, Upper Saddle River, NJ, USA

Matthew, N. & Stones, R. (2004). *Beginning Linux Programming*, 3rd edition, Wiley publishing Inc., Indianapolis, Indiana, USA

http://www.maxbotix.com/uploads/LV-MaxSonar-WR1-Datasheet.pdf, - Maxbotics Inc., *Accesed on: 2010-10-20*

Nobutaka, I. (1990). Agricultural robots in Japan, *IEEE International Workshop on Inteligent Robots and Systems – IROS 90*, pp. 249-253

Stajnko, D.; Rakun, J. & Blake, D. (2009). Modelling Apple Fruit Yield Using Image Analysis for Fruit Colour, Shape and Texture, *European journal of horticultural science*, vol. 74, pp. 260-267

http://fasterdata.es.net/TCP-tuning/linux.html - TCP Tuning Guide ver. 1.0, *Accesed on: 2010-10-20*

Thompson, J. F.; Stafford, J. V. & Miller, P. C. H. (1991). Potential for automatic weed detection and selective herbicide application, *Crop Protection*, vol. 10, pp. 254-259