Design and Implementation of a Real-Time Tracking and Telemetry System for a Solar Car

Elie N. Mambou, Theo G. Swart, Hendrik C. Ferreira, A. R. Ndjiounge
Telecommunication Research Group
University of Johannesburg
P.O. Box 524, Auckland Park, 2006, South Africa
Email: elymambou@yahoo.fr, {tgswart, hferreira, arrichard}@uj.ac.za

Abstract—It is very critical for solar racers to achieve a real time tracking of their solar vehicle during any race. The system is made of the real time visualisation of the video feedback of the car from the escort vehicle, the location of the car on the google maps as it is racing and its velocity. The real time video feedback of the race could be retrieved from the cloud for broadcasting purposes. The main advantage of this system is the wireless link communication which was achieved for distance transmission, coupled to a very reliable and robust Desktop Application for data sensor display purposes.

Index Terms— RaspberryPi, Graphical User Interface, Global Positioning System, Inertial Measurement Unit, Micro Controller Unit, Controller Area Network

I. INTRODUCTION

As fuel cost continues to increase, many questions arise concerning the future of combustible resources that is why numerous automotive manufacturers are developing hybrid vehicles. At the same time many research are been conducted in alternative fuel vehicles [1].

The South Africa Solar Challenge (SASC) is a biannual event that happening in South Africa and organised by the company called ‘Sasol’, which is one of the biggest fuel extractor firm in South Africa (SA). The aim of this challenge in to promote green energy, especially the solar energy which nowadays, constitutes a reliable alternative source of energy. During the challenge, solar racers are called to compete along a well defined route in SA (See Fig 1). The winner of the race is the team which solar car has travelled the highest distance during the challenge [2].

During a solar challenge race, the solar vehicle is followed by an escort car. The role of the escort car is to observe and analyse the performance of the solar vehicle in order to monitor and optimize the race strategy. With the purpose of attracting a lot of visitors and raising awareness about the solar energy during the SASC, the monitoring system should include a real time video feedback of the solar-vehicle which can be accessible on the website, this is to allow solar energy followers to follow the performance of a solar car during the race. Furthermore a Global Positioning System (GPS) and Inertial Measurement Unit (IMU) system is needed to keep track with accuracy the motion of the solar car during the race.

This research was done to improve the telemetry system of the University of Johannesburg (UJ) solar-powered car that was essentially based on Global System for Mobile (GSM) communication, designed and provided by a base station network.

The UJ solar team after participating in the 2012 SASC with the famous Ilanga 1 Solar Powered Car, opted to build a more advanced car, called Ilanga II (“Ilanga” means “Sun” in isiZulu) for future races. This project involved a lot of students from different fields especially industrial designers, electrical and electronic engineers and mechanical engineering students under the mentor-ship of some lecturers and companies.

Many research have been conducted by other solar teams around the world on the telemetry and tracking system.

The team 4431 solar car telemetry system is based on the XStream OEM Radio Frequency (RF) module. This device allows an outdoor line of sight range of 7 miles, has a frequency of 900 Mhz and receiver sensitivity of $-110$ dBm. The sender is on the solar vehicle while the receiver is sitting on the escort car. This allows the transmission of data via a wireless link. The main concern with device is the throughput data rate of 9600 bps which may not support the data congestion during transmission. The Labview environment was choosen for GUI [3].

The center for product design and manufacturing (CPDM) of University Malaya presented almost the same telemetry design as before. The sensors are connected to I/O modules, the real time controller is responsible to process and send data to the Xstream OEM RF 900 MHz modules through the RS232 serial port. The data was saved into cRIO as back up. The
receiver was connected to the computer via a USB-RS232 cable for data acquisition [4].

The two previous systems are based on NI Labview components which are very expensive and allow a small throughput data rate and do not achieve tracking of the solar car. The system presented in this paper is based on cost-efficient electronic features, with a total implementing cost of less than 150 $ US, which is very low comparing to other systems. This is very indicated for small teams or beginner solar racers. The main advantage of this system is the achievement of video feedback transmission via wireless link which is a very huge data which makes the tracking more interesting, coupled to the live location of the solar vehicle on google map.

II. TRACKING SYSTEM DESIGN AND IMPLEMENTATION

The proposed tracking system is divided into 3 subsystems:

- Sensor board (SB): Its goal was to sense the position of the car during the race and sending this data every 1 second to the chase car via wireless. The SB was designed in Circuit design software called Altium Designer. It was important to design the SB from scratch because of low power consumption purposes as it is needed to dissipate as little power as possible from the batteries which is not the case for most of the On-board computers.
- Camera module (CM): The CM was mounted onto the Raspberry Pi (RPi) board which was programmed to send the video feedback of the solar vehicle to the chase car.
- Desktop Application (DA): It is designed to provide a nice display of the video feedback as well as vehicle position on the map from a remote computer.

![Fig. 2. Overview of the Tracking System](image)

A. Sensor board design

The sensor board works as follows: The board is placed in the solar vehicle in a static position. The board is powered through Controller Area Network (CAN) bus which is connected to the battery of the vehicle supplying a voltage of 12 V. The microcontroller unit (MCU), PIC32M795F512L from Microchip technologies was used to synchronises data coming from the GPS chip and transmit them to the chase car via a wireless link. The purpose of some features on the sensor board and how they are interfaced with the MCU are detailed as follow.

- Voltage regulator circuit: It has a role of maintaining and regulating voltage level through the circuitry. In our case there are 2 voltage regulators. The board is powered from 12 V DC, the first regulator (LM2596SX-5.0) step-down the input voltage to 5 V (input voltage of USB port) and the second regulator (LM2596SX-3.3) step-down to 3.3 V (input voltage of GPS receiver, SD card socket, MCU).
- PIC32 Connection: The basic connection for the PIC32 includes all VSS, AVSS pin to be grounded, similarly all VDD, AVDD pins should connected to 3.3 V. The ceramic capacitor on the board with a value of 100pF are used to reduce power fluctuations throughout the circuitry. The PIC32 has an internal crystal of 8 MHz which is relatively low, a crystal oscillator of 28 MHz was added to provide more stable oscillation. The MCLR pin (Master CLear Reset) was connected to a push button and a resistance of 500 ohm was placed in between to limit the current flowing through the line for safety purposes.
- ICD (In-Circuit Debugger) connection: This is a port on the board which enables the programming of the chip, in our case an In-Circuit debugger 3 (ICD 3) was used to debug the chip.
- GPS module: The NEO-6G GPS receiver was used. The most important connection on the GPS schematic is the transmission pin (TXD) of the GPS receiver which is connected to a reception pin (RXD) of the MCU. Because of the short range of the GPS receiver which could not give any reading indoors therefore a GPS antenna with a signal-to-noise ratio (S/N: 308400) was mounted on the board to increase the accuracy in data.
- Secure digital (SD) card connection: The SD card is connected to the MCU via a SD card socket. The purpose of the SD card was to save the positions of the vehicle in a text file given by the GPS sensor.
- Wireless module: A Wireless transceiver called XBee S2 was embedded in the SB to achieve the transmission of data to the remote laptop. The protocol used was 802.15.4 at 2.4 GHz.
- USB Port: This port was an alternative way of powering the sensor board from 5V in case that a 12V power supply (battery voltage) was not available.

Fig. 3 presents the sensor board being programmed using MPlabX IDE. With the board connected to a 12V DC power supply from CAN bus and programmed via an ICD3 debugger.

B. Camera Module subsystem

Fig. 4 shows a snapshot of a digital clock video feedback taken by the RPi camera module mounted on the RPi board and transmitted via wireless to another computer through a unique internet protocol (IP) address and a communication port. As it can be seen on the snapshot, there was a delay.
of about 4 seconds in the transmission. This delay was due to the streaming of the video from receiver side through the port pipeline. The RPi was dotted with a WiFi dongle with the following properties: 2.4 Ghz, 150 Mbps, 802.11 b/g/n Wifi.

![Sensor board](image)

**Fig. 3. Sensor board**

The following code from a script file was parsed to perform a wireless transmission of the video feedback to the desktop application:

```
sudo apt-get update
sudo apt-get upgrade
raspivid -width xx -height xx -t xx
-framerate xx
-output xx - || cvlc -vvv
stream:///dev/stdin -sout
# rtp {sdp=rtsp://portNumber }
```

Raspivid is the library for camera functions in RPi that includes the width, the height, the number of frames per second. On the other hand the output of the video is sent to a remote computer via a port number through a VideoLan Converter Media Player (VLC) File Transfer Protocol (FTP) pipeline. However on the remote computer, in order to access the video, the IP of the RPi is required as well as the same port number of communication.

**B. Remote desktop application subsystem**

![Snapshot of RPi camera transmitting video](image)

**Fig. 4. Snapshot of RPi camera transmitting video**

On the left-hand, there is the video feedback retrieved from the RPi.
- **Stream button**: Open the connection between the remote computer and the RPi board through the same port number.
- **Play button**: play of the real-time video.
- **Pause button**: pause the incoming video.
- **Browser button**: Local Browsing of any video for testing purposes.
- **Stop button**: stop the video.
- **Volume gauge**: adjust the volume of the streamed video.

On the right-hand, the application allows snapshots saving from the live video. The button “capture” takes the snapshot of the current frame in the video viewer and the button “save” stores the snapshot taken in picture format in a browsed directory.

![Remote desktop application view of transmitted video](image)

**Fig. 5. Remote desktop application view of transmitted video**

**C. Desktop application**

This subsystem was based on the effective reception of these data, evaluating the delay in streaming and saving data. The desktop application was called “UJ Solar Car Video Tracker”. This application was implemented using “C sharp” programming with windows presentation foundation (WPF) in the Microsoft visual studio 2012 environment.

**III. Results**

**A. Camera subsystem**

The camera system was successfully implemented as follow: A camera module was mounted onto the RPi, the commands

```
sudo apt-get update
sudo apt-get upgrade
raspivid -width xx -height xx -t xx
-framerate xx
-output xx - || cvlc -vvv
stream:///dev/stdin -sout
# rtp {sdp=rtsp://portNumber }
```

sudo apt-get update and sudo apt-get up-grade were executed in the Linux environment to update the firmware of RPi and install all necessary libraries. The library called Raspberry Pi Video (Raspivid) was used to interact with the camera module.

The following code from a script file was parsed to perform a wireless transmission of the video feedback to the desktop application:

```
sudo apt-get update
sudo apt-get upgrade
raspivid -width xx -height xx -t xx
-framerate xx
-output xx - || cvlc -vvv
stream:///dev/stdin -sout
# rtp {sdp=rtsp://portNumber }
```

Raspivid is the library for camera functions in RPi that includes the width, the height, the number of frames per second. On the other hand the output of the video is sent to a remote computer via a port number through a VideoLan Converter Media Player (VLC) File Transfer Protocol (FTP) pipeline. However on the remote computer, in order to access the video, the IP of the RPi is required as well as the same port number of communication.

**B. Remote desktop application subsystem**

![Remote desktop application view of transmitted video](image)

**Fig. 5. Remote desktop application view of transmitted video**

On the left-hand, there is the video feedback retrieved from the RPi.
- **Stream button**: Open the connection between the remote computer and the RPi board through the same port number.
- **Play button**: play of the real-time video.
- **Pause button**: pause the incoming video.
- **Browser button**: Local Browsing of any video for testing purposes.
- **Stop button**: stop the video.
- **Volume gauge**: adjust the volume of the streamed video.

On the right-hand, the application allows snapshots saving from the live video. The button “capture” takes the snapshot of the current frame in the video viewer and the button “save” stores the snapshot taken in picture format in a browsed directory.
C. Sensor board subsystem

Concerning the sensor board, Fig. 6 presents the programming setup of the SB which is connected to a 12 V power supply and the ICD3 debugger. The positions of the car by means of GPS coordinates were logged in a text file inside the SD card.

From the desktop application point of view, Fig. 7 presents the position of the car on the Google map from the Graphical User Interface (GUI). This map was updated after every second to display the current position of the car and the coordinates were shown on the side on the application.

However the positions of the car were saved after every second from the SD memory of the sensor board located on the car. This was achieved before the data got transmitted to the escort remote application to be mapped.

The snapshot of the file containing some positions of the car was illustrated in Fig. 8.

IV. CONCLUSION

An approach to real-time tracking and telemetry system of a solar-powered car was proposed in this paper. The camera system was implemented successfully in the sense that the video feedback was sent over wireless through a static IP address and a port to a remote computer with a delay of around 3 seconds. This is a great achievement as a bridge was established between a Linux and a Windows platform to transmit huge amount of data. The SB was designed, manufactured and assembled successfully. The GPS sensor u-blox NEO-6G placed on the SB was able to send GPS coordinates through the SB microchip, PIC32MX575F512L. The data from that sensor was received after every second which is the frequency at which data was received from the GPS chip. The data was retrieved from a remote computer as well as the video feedback. Finally the desktop application was done, the video feedback from the camera module of the RPi was received as well as the positions of the car.

V. FUTURE WORK

Nowadays, most of the camera surveillance systems are wired connected, which increases the cost of implementation due to the distance of communication; therefore with our camera subsystem, a real-time camera surveillance can be installed in houses or companies or anywhere; all camera view within the system can be monitored from a remote server computer. The desktop application of the system which is very user friendly, after being installed on a windows computer, can communicate with any Linux based electronic board via a unique port and a broadcast IP address given by the network.
Furthermore our proposed camera system can incorporate GPS data in the sense that every camera view is retrieved on the remote server computer with accurate GPS coordinates and time and therefore can be located on the geographic map. This provides an modern camera surveillance and security management system.

ACKNOWLEDGMENT

The authors would like to thank the Solar team 2014 of the University of Johannesburg with their sponsors and mentors.

REFERENCES