Paper: ASAT-15-097-CT

15th International Conference on
AEROSPACE SCIENCES & AVIATION TECHNOLOGY,
ASAT - 15 – May 28 - 30, 2013, Email: asat@mtc.edu.eg ,
Military Technical College, Kobry Elkobbah, Cairo, Egypt,
Tel: +(202) 24025292 –24036138, Fax: +(202) 22621908

System Design and Realization of an Autonomous Unmanned Ground Vehicle using GPS-Based Navigation

A. Haytham*, A. Wasal†, Y. Z. Elhalwagy‡, and M. Elfaramawy§

Abstract: Unmanned vehicles have been an active area of research and development over the past three decades. Unmanned Ground Vehicles (UGVs) are used in both civilian and military applications all over the world. The design of the ground control station hardware and software can become very complicated depending on the UGV mission and the autonomous guidance required. This paper introduces an autonomous system capable of guiding a small UGV via GPS. Different control methods are used including radio control (RC), joystick, and autonomous control. The GPS navigation loop provides continuous and reliable navigation data to the waypoint guidance algorithm and its control loop which is responsible for autonomous navigation. This control loop computes the guidance commands to follow the waypoint scenarios encountered and generates actuator signals for steering and speed control. The guidance, navigation, and control (GNC) algorithm was implemented within an embedded processor which communicates with the ground control station (GCS) over a wireless channel. The real-time test results indicate that the vehicle can reliably perform autonomous guidance in different scenarios.

Keywords: UGV, autonomous system, navigation, guidance, control, GPS, GNC

Introduction

Unmanned ground vehicles (UGV) are wheeled robots capable of performing tasks without human intervention. Human intervention, however, may be used/required in some situations. A UGV is equipped with sensors to interact with its surrounding. In order to move autonomously, a UGV requires a navigation, guidance, and control system. The accuracy of navigation, guidance and control depends on the performance of the sensors [1]. Sensors commonly utilized in UGVs include global position system (GPS) receivers, inertial measurement units (IMU), compasses, and visual sensors [2]. Multi-sensor data fusions and sensor integration significantly improve both the overall system performance and the navigation accuracy. This improvement, however, comes at the expense of increased system cost and complexity.

This paper presents a simple design and low cost, realization of an autonomous system for guiding a small UGV. The system employs a commercially available GPS receiver in the feedback closed loop guidance, navigation, and control (GNC) process, and an embedded

* Egyptian Armed Forces, Egypt.
† Assoc. Prof, Computer Eng. Dept, Cairo University, Egypt
‡ Egyptian Armed Forces, Egypt.
§ Egyptian Armed Forces, Egypt.

ARM processor [3] for executing the GNC algorithms. The advantages of using a sole sensor are the low cost, simplicity, and system miniaturization [4]. The main advantage of using GPS is that the data collected is independent of previous data samples. Therefore, errors in localization do not accumulate over time. However, the accuracy and precision of the GPS receiver are affected by its surroundings, the number of satellites available, and the GPS update rate [5], [6]. The features and capabilities of the ARM processor meet the hardware, software, real-time processing, and memory requirements of the GNC process.

Autonomous navigation encompasses two main operations, namely, vehicle localization and motion planning [7].The vehicle localization operation refers to determining the position and orientation of the vehicle. It is performed by the GPS receiver, which determines the position and speed of the vehicle, and provides continuous and reliable navigation information to it. The motion planning operation refers to deciding on the next move of the vehicle, in order to execute the required mission. It is performed by the guidance algorithm, which computes the next vehicle movement based on the current geometric constraints.

This paper is organized as follows. Section 2 describes the hardware architecture of the autonomous system. Section 3 describes the hardware design of the different control methods. Section 4 discusses the software structure, including the GNC algorithm. Section 5 outlines the message protocol between the ground control station (GCS) and the microcontroller. Section 6 describes the functionality of the GCS. Experimental results are presented in Section 7, and the conclusion is given in Section 8.

Hardware System Architecture

A block diagram of the autonomous vehicle is shown in Fig. 1. It consists of three subsystems, the vehicle platform, avionics, and GCS [8].

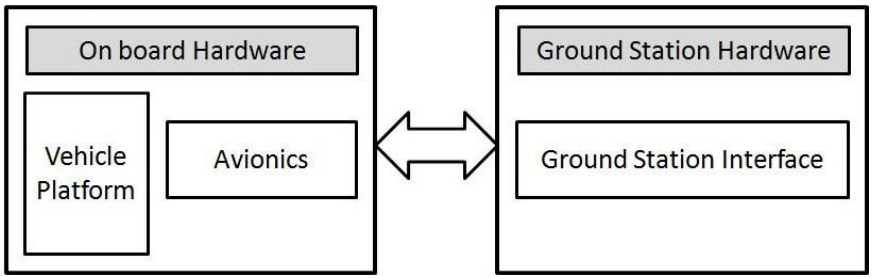

Fig. 1 The autonomous vehicle subsystems

The vehicle platform consists of the vehicle dynamics, RC servos, an RC receiver, batteries, and a speed control system. The avionics consist of an ARM7 microcontroller (LPC2378 series [9] mounted on an MCB2300 board from Keil [10]), a GPS receiver, a radio modem, and a bypass circuit. The GCS includes a laptop computer, a radio modem, a joystick, and an RC transmitter.

Figure 2 illustrates the architecture of the autonomous vehicle system, showing the components of each of the three subsystems and their interconnections. The microcontroller board runs the main control software, and interacts with several components during normal operation.

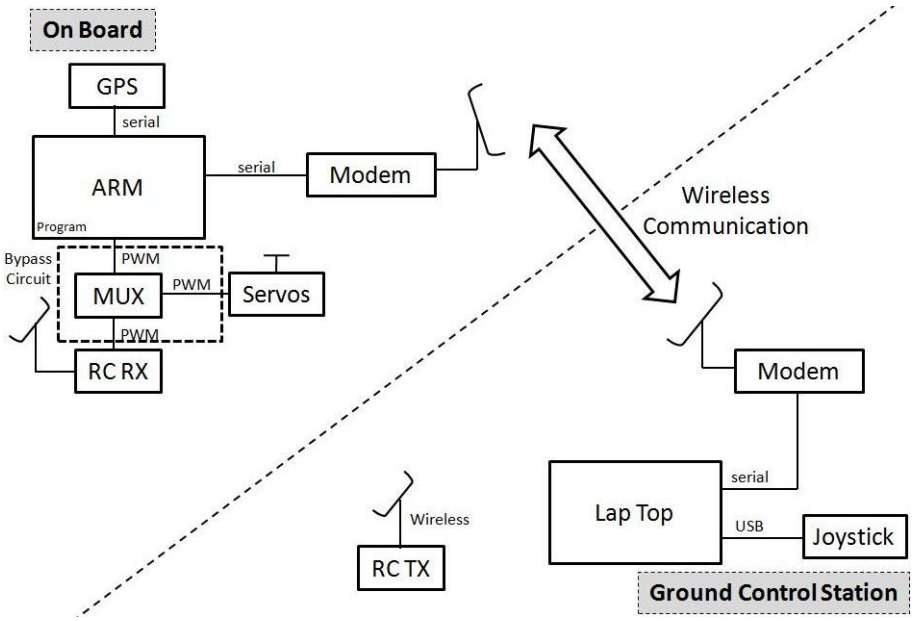

Fig. 2 Autonomous system architecture

The Bypass circuit is an essential component of the on board avionics during autopilot development [8]. Should a problem arise during testing that puts the vehicle in danger, the human pilot can immediately take control of the vehicle using a toggle switch on the RC transmitter. The human pilot manually inspects the vehicle, in order to examine the problem that occurred during the mission. This approach to autopilot development allowed testing new algorithms and hardware with minimal risk to the vehicle.

System Work Flow

Figure 3 illustrates the structure of the on board components of the autonomous vehicle. There are three possible methods for vehicle control, namely; radio control, joystick, and autonomous control. This is facilitated by the bypass circuit, which provides an additional path from the RC receiver.

There are two modes of control, Pilot-in-Control (PIC) and Computer-in-Control mode (CIC) [7]. In PIC mode, the vehicle is controlled by a human pilot through the RC transmitter, whereas in the CIC mode, the microcontroller has control of the vehicle. The CIC mode can be further classified into CIC remotely piloted vehicle (CIC-RPV); where the vehicle is controlled by a joystick at the GCS, and CIC unmanned ground vehicle (CIC-UGV); where the vehicle is autonomously controlled according to a pre-designed mission.

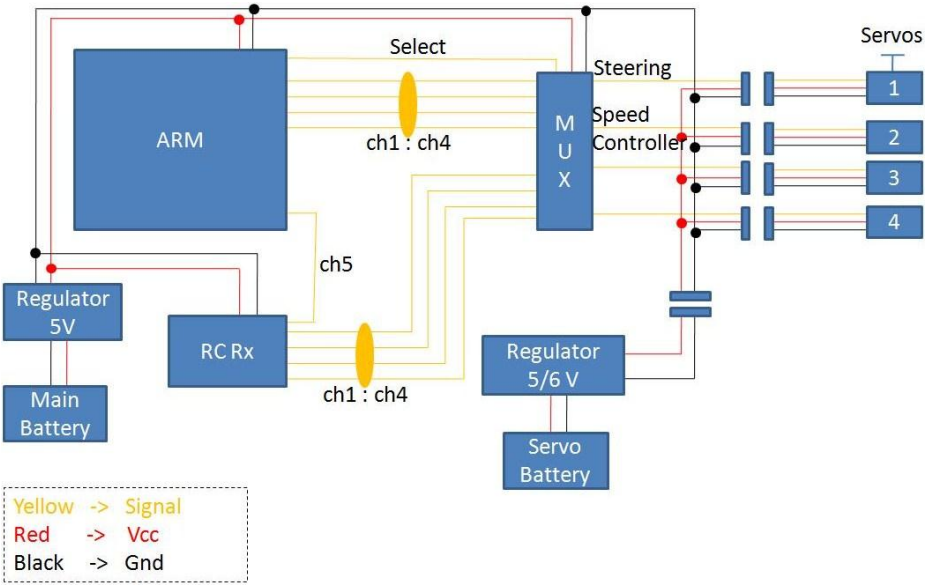

Fig. 3 Wiring of the on board components of the autonomous system

The PIC Mode

The RC transmitter at the GCS allows a human pilot to control the vehicle. It is essential during the testing, where the pilot can take over from the autopilot in an emergency, in order to prevent accidents.

Figure 4 illustrates this mode of control. Channel 5 (Ch 5) is decoded separately in the ARM board and used to control the MUX. If the PWM signal of Ch 5 is changed, it generates an interrupt. Thus, the main loop is stopped and the interrupt service routine (ISR) for this interrupt is executed. Then, the main loop resumes after the last executed command.

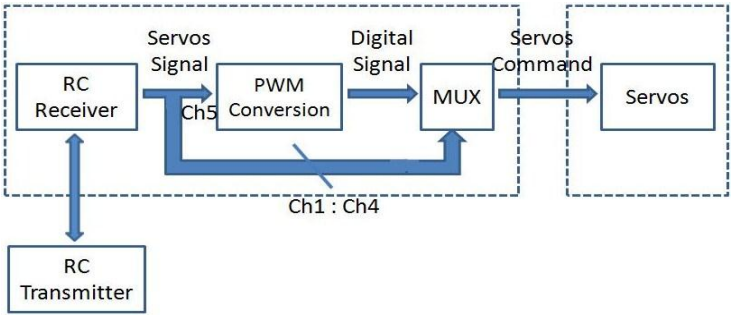

Fig. 4 PIC mode work flow

If the Ch 5 pulse falls below a pre-specified threshold of 1500 µsec, the ARM board switches the multiplexer of the bypass circuit to connect the RC receiver servo outputs (Ch 1 to 4) directly to the servos thereby entering the PIC mode. A human pilot controls the vehicle through the 72 MHZ control link. If the Ch 5 pulse rises above the threshold, the multiplexer connects the microcontroller servo output channels to the servos, and hence enables the CIC mode.

The CIC-RPV Mode

The vehicle is controlled by a joystick which is connected to the GCS via a USB cable. Communication with the vehicle occurs using the radio modem. Therefore, the control range is determined by the radio modem range. In this mode, vehicle control depends on the gauges and the map at the GCS.

The toggle switch of the RC transmitter is on the position of the CIC mode. Since the default control mode is CIC-UGV, the GCS sends a command to the autonomous system, which generates a UART interrupt to switch the mode to CIC-RPV.

As shown in Fig. 5, the received telemetry messages, which are displayed on the gauges and the map, help the pilot guide the vehicle. The movements of the joystick are captured by the GCS. The GCS then generates messages to the microcontroller containing the appropriate servo number and the required movement to be executed. The microcontroller generates PWM based on the messages delivered to the RC servos.

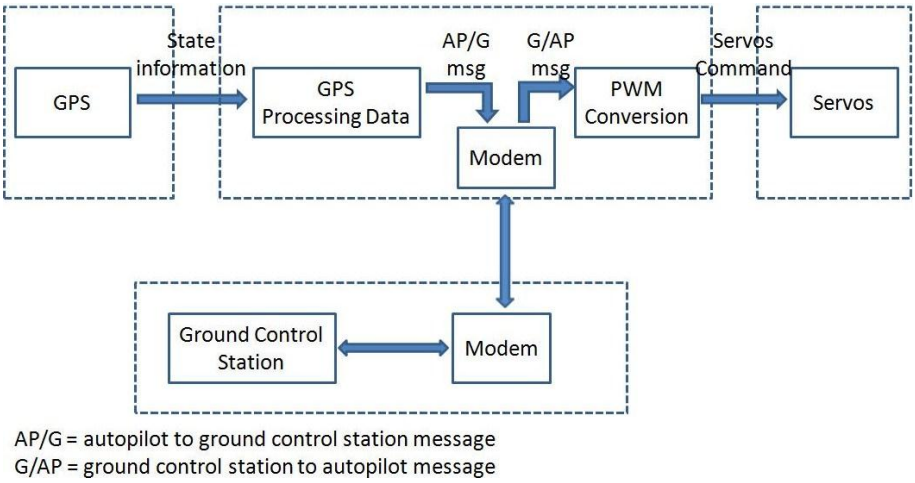

Fig. 5 CIC-RPV mode work flow

The CIC-UGV Mode

When the toggle switch position; of the RC transmitter, is on the CIC mode, the autopilot takes over the control of the vehicle.

As shown in Fig. 6, the GPS feeds the navigation function with the current position, orientation, altitude, and speed. This data, along with the mission waypoints, are input to the guidance algorithm. After the guidance algorithm is executed, it outputs the target true angle to the waypoint and the target speed value. The PID controller uses these two values to determine the required servo angle for the next vehicle move. The servo moves the vehicle wheel, changing the current position and orientation of the vehicle.

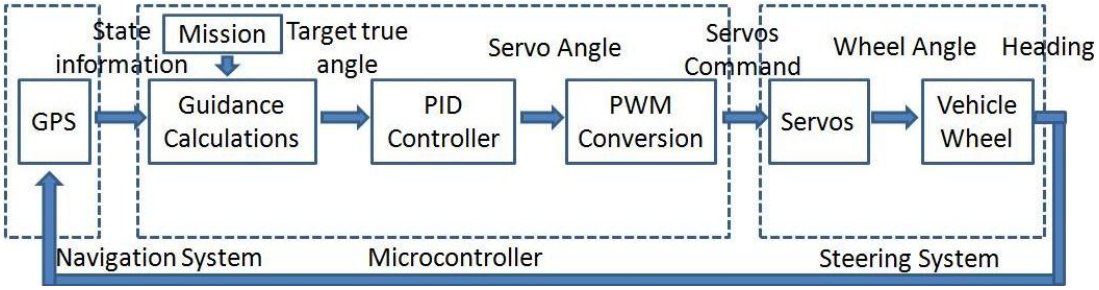

Fig. 6 CIC-UGV mode work flow

The new position and orientation are fed back to the autopilot. This loop is iterated until the vehicle reaches within 40 cm of the target waypoint (which is deemed acceptable arrival at that waypoint). The next waypoint in the mission then becomes the new target waypoint. The above process is repeated until the vehicle reaches the last waypoint, where the vehicle stops indicating mission completion.

Software Structure

The autonomous system software is divided into three modules, as shown in Fig. 7. First, the pre-main section is executed only once when power is switched on. Second, the autopilot code is executed in the main loop, which runs continuously. Finally, the ISR module runs in parallel with the main loop.

The waypoints of the required mission are stored in the pre-main module. This information is passed to the navigation function in the main loop. The navigation function is responsible for gathering and processing the GPS data. The GPS update rate is 200 msec. Moreover, it reads the National Electrical Manufacturers Association (NEMA) messages from the serial port.

The guidance and control function executes the guidance algorithm and determines the control algorithm. The PID controller manages two separate control loops for speed hold and steering adjustment. It generates the appropriate PWM to the servos for each new movement.

The telemetry function constructs telemetry messages and sends them to the GCS for real-time monitoring and mission analysis.

The following subsections are devoted for illustrating the proposed waypoint guidance algorithm associated with the geometrical relations and the PID control algorithm for both steering and speed control loops.

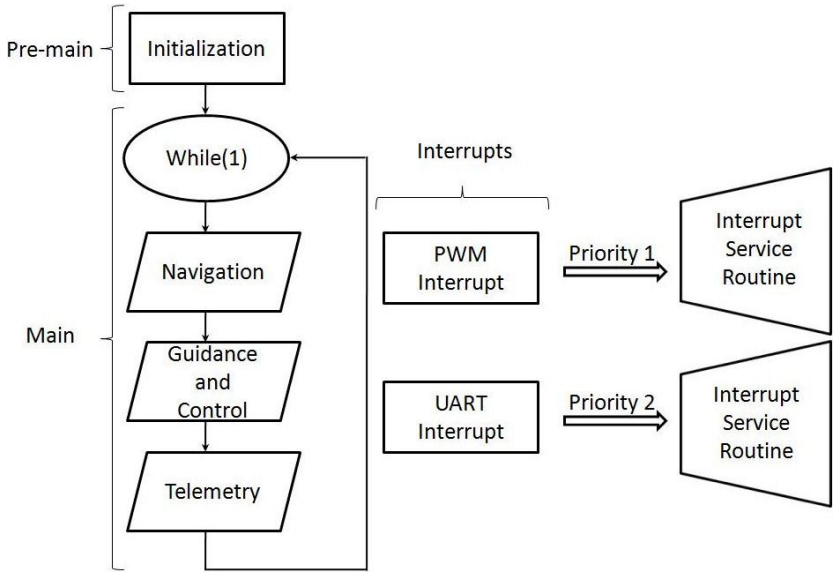

Fig. 7. The structure of the autonomous system software

The Guidance Algorithm

The guidance loop calculates the track angle error (α).α is the off-course angle, which is the difference between the calculated bearing to destination angle (b2dest) and the given course angle from the GPS (β). Figure 8 illustrates the parameters and geometry of the algorithm [6], when the vehicle is moving towards waypoint (WP1). The GPS determines the current vehicle position, i.e., its latitude (Lat) and longitude (Long), and the course angle, which is the heading angle measured from the north. Given the target waypoint (WP1) coordinates, Δx and Δy are calculated using:

$$\Delta x = Long\ WP1 - Long\ Vehicle\ GPS \quad (1)$$

$$\Delta y = Lat\ WP1 - Lat\ Vehicle\ GPS \quad (2)$$

The radius to destination (r2dest), or the distance between the vehicle position and the target waypoint, is computed using the Pythagorean Theorem as:

$$r2dest = \sqrt{\Delta x^2 + \Delta y^2} \quad (3)$$

Therefore, the b2dest angle computed using:

$$b2dest = \sin^{-1}\left(\frac{\Delta x}{r2dest}\right) \quad (4)$$

Finally, the track angle error is calculated using:

$$\alpha = b2dest - \beta \quad (5)$$

The angle α constitutes the guidance command, which is input to the control loop, which in turn aims at minimizing α.

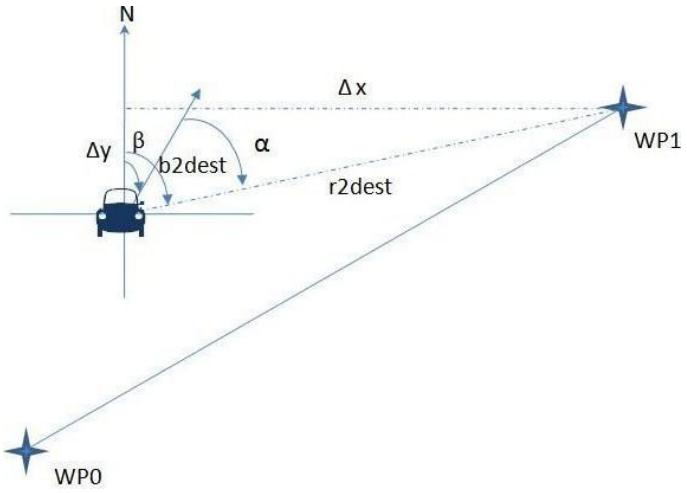

Fig. 8 Computation of the tracking angle error, α

The Control Algorithm

The PID controller is used to steer the vehicle in order to follow the mission path [11]. Figure 9 illustrates the steering control loop. The tracking error angle is input to the PID controller, which determines the required servo angle. The servo angle ranges from -45° to 45°. The servo moves the vehicle wheel, changing the current position and orientation of the vehicle. The wheel angle and the vehicle velocity are inputs to the car model.

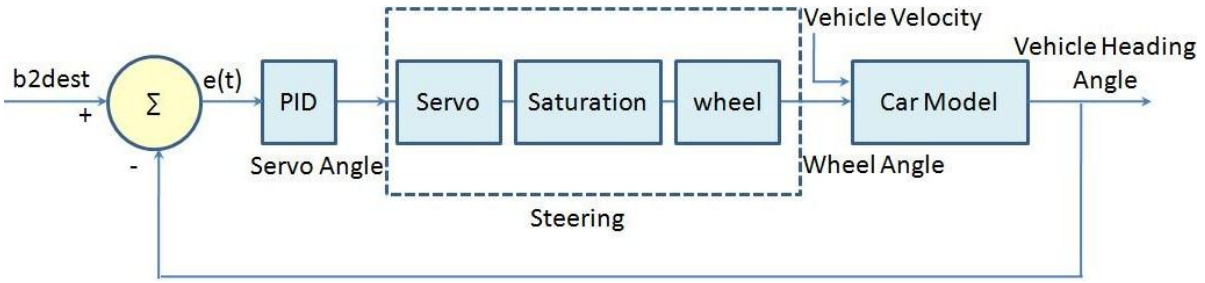

Fig. 9 The steering control loop

The PID controller also controls the vehicle speed, as shown in Fig. 10. It is responsible for maintaining a constant speed throughout the entire mission.

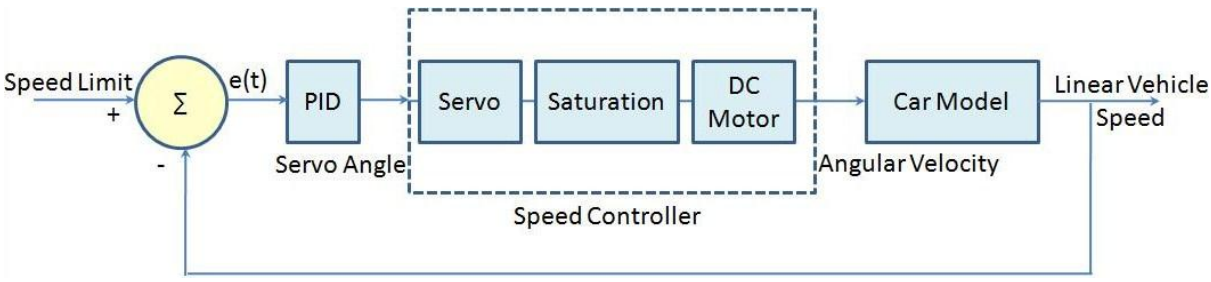

Fig. 10 The speed control loop

The PID controller gains were tuned manually using MATLAB's genetic algorithm toolbox. The genetic algorithm is initialized manually to reduce the search time.

The Message Protocol

Different message types are exchanged between the microcontroller and the GCS. At the beginning of the connection, the GCS sends a mission scenario message to the microcontroller. The microcontroller sends telemetry messages to the GCS throughout the entire mission at a frequency of five messages per second.

The message format consists of a header, a message body, and a checksum. The checksum is used for validation, and is based on the longitudinal redundancy check [12]. Messages can have a fixed or a variable length, depending on the message type.

Figure 11 illustrates a sample RPV-UGV message sent from the GCS to the microcontroller. This message has a fixed length of five bytes. The first byte is the message header, the second byte is the servo ID, the third and fourth bytes are the required servo value, and the last byte is the checksum.

Fig. 11 A sample RPV-UGV message from the GCS to the microcontroller

In the CIC-RPV mode, the third bit in the servo ID byte is set to one and the first two bits are the servo ID. The servo value bytes carry the joystick commands to the microcontroller in order to generate PWM according to the servo value. On the other hand, in the CIC-UGV mode, the third bit is set to zero. The rest of the message is ignored.

The Ground Control Station

The GCS has been developed by Visual C# language for the monitoring and visualization of the vehicle parameter variation. The GCS interface is shown in Fig. 12. The GCS performs four main functions [13]:

- Mission planning: The GCS interface allows the user to design the required mission and send it to the microcontroller for execution. A mission can be saved and executed at a later time.
- Data manipulation: The GCS receives real-time data. The received data is processed and displayed to help the user to analyze the mission.The data may also be archived during or after the mission, for later analysis.
- Mission control: The GCS allows the user to send commands to the autopilot at runtime using the GCS interface, as well as the joystick.
- Mission play back: The GCS interface can play back archived data for the purpose of analysis.

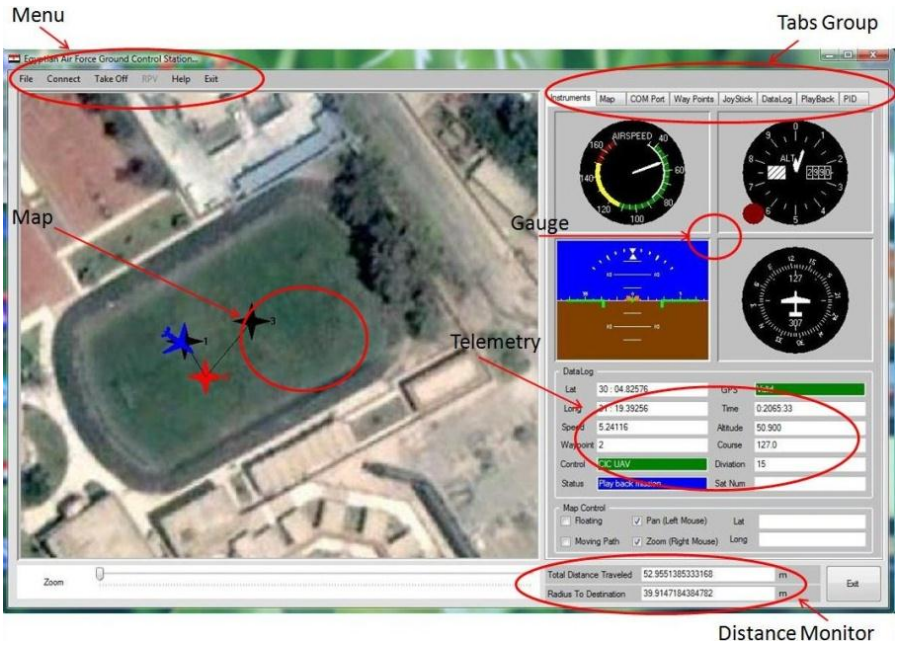

Fig. 12 The ground control station interface

Experimental Results

The mission scenario is designed using the GCS interface and sent to the microcontroller. The mission scenario used in the reported experiments consists of three waypoints, WP1, WP2, and WP3.

Figure 13 shows the experimental vehicle (left) and the onboard autonomous system (right). The vehicle moves at a speed of 3 knots (5.556 km/h) towards WP1. When the vehicle reaches within 40 cm of WP1 (which is deemed acceptable arrival at a waypoint), it moves towards WP2. This process is repeated until the vehicle reaches the last waypoint, WP3.

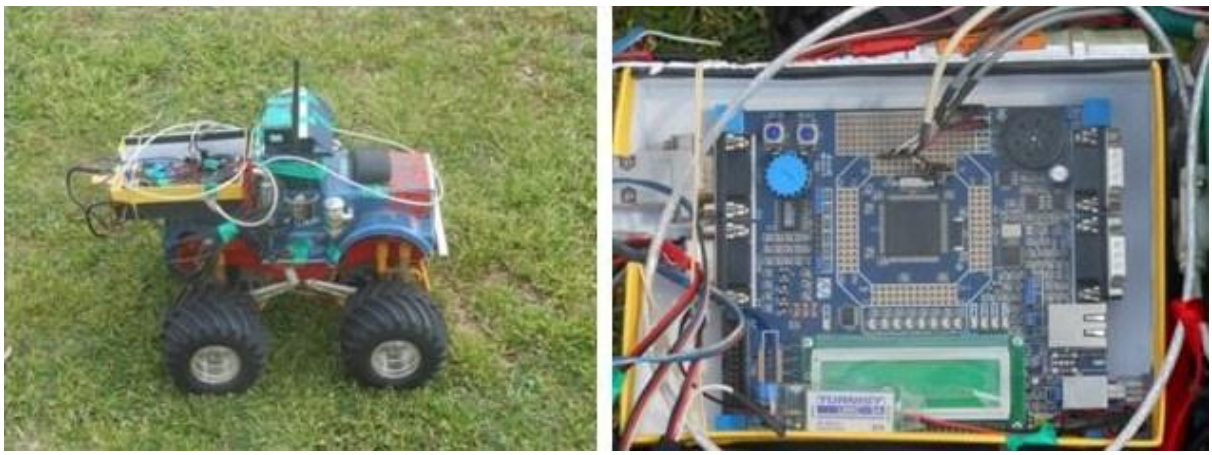

Fig. 13 Experimental vehicle and the onboard subsystem

Two experiments were performed with different PID gains in the speed control loop. Table 1 lists the PID gains used in the two experiments. k_p is changed from 0.2 to 1 in the second experiment. Figure 14 illustrates the mission scenario in experiment 1, shown using the green straight lines. The actual vehicle position is shown using the blue curve. Acceptable arrival at a waypoint is established when the vehicle reaches within 40cm of that waypoint (See the

three circles). The left most (red) part of the curve represents the initial part of the mission where the vehicle is in the PIC mode. The remaining (blue) part of the curve represents the position of the vehicle during the rest of the mission, where the vehicle is in the autonomous movement (CIC-UGV) mode. Figure 15 illustrates the vehicle path in experiment 2. It is clear that in experiment 2, the vehicle position is improved throughout the entire mission.

Table 2 lists the least squared error of the vehicle position during each leg of the mission in both experiments. The table indicates that the set of PID gains used in experiment 2 results in less error. Figures 16 and 17 compare the actual vehicle speed in knots (red) to the reference constant speed (blue) in experiments 1 and 2, respectively. The green line represents the target waypoint number. The figures clearly show how the vehicle speed is improved in experiment 2. This fact is also evident from Table 3, which compares the least square error of the vehicle speed in both experiments.

Table 1 Steering and speed PID gains for experiments 1 and 2

	Experiment 1			Experiment 2		
Control	k_p	k_i	k_d	k_p	k_i	k_d
Steering	1.111	0.0555	5.555	1.111	0.0555	5.555
Speed	0.2	0.01	1.0	1	0.01	1.0

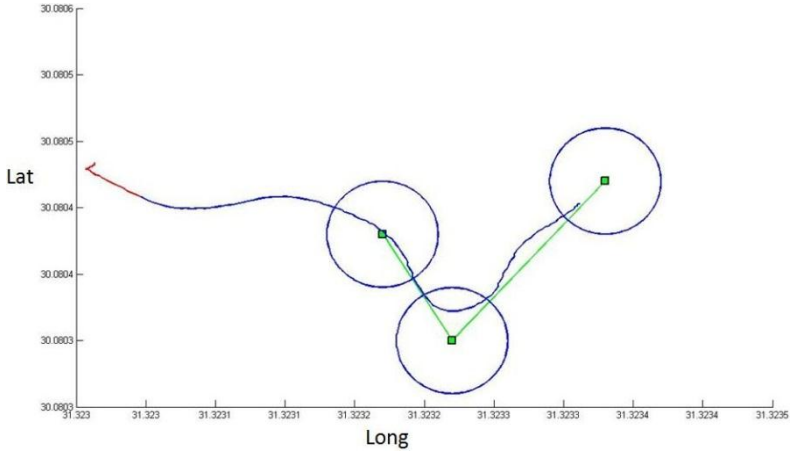

Fig. 14 The mission scenario (green) and the actual (blue) vehicle path in experiment 1

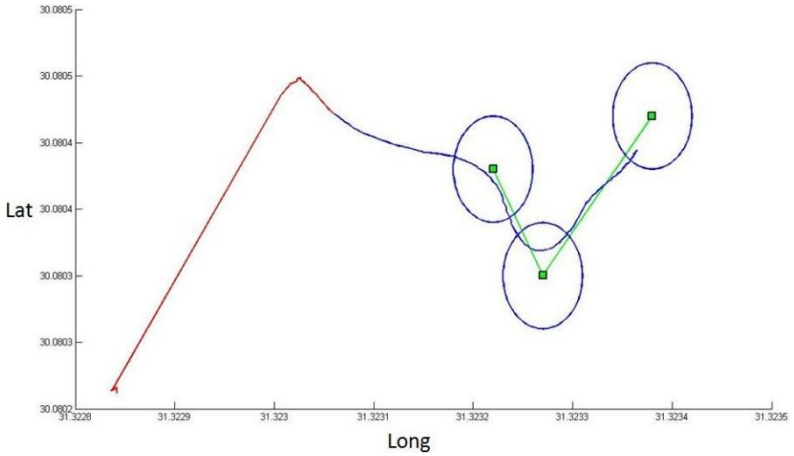

Fig. 15 The mission scenario (green) and the actual (blue) vehicle path in experiment 2

Table 2 Vehicle position least squared error for experiments 1 and 2

	Experiment 1			Experiment 2		
Target Leg	WP1	WP2	WP3	WP1	WP2	WP3
Least square Error	13370	7878	60170	2888	6810	59896

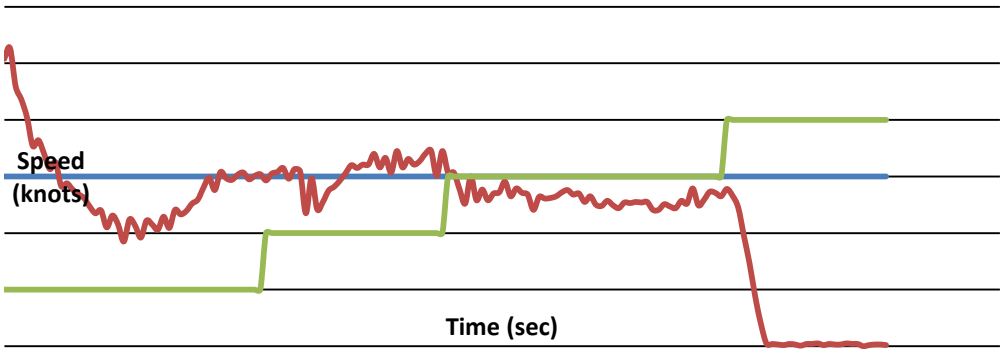

Fig. 16 The actual vehicle speed (red) versus the reference speed (blue) during the three legs of the mission in experiment 1

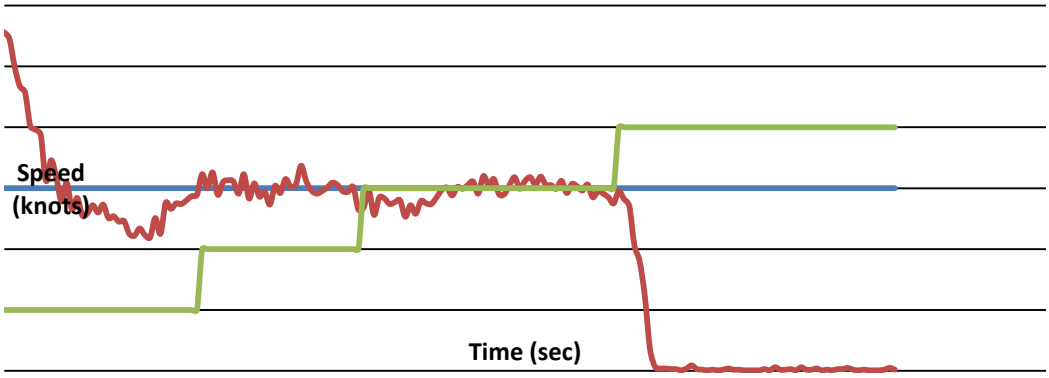

Fig. 17 The actual vehicle speed (red) versus the reference speed (blue) during the three legs of the mission in experiment 2

Table 3 Least squared error for experiments 1 and 2

	Experiment 1	Experiment 2
Least square error	38.73	33

Conclusion

This paper discussed the implementation of an autonomous system capable of guiding a small UGV equipped with a GPS receiver. The implementation allows three control modes, namely, RC transmitter, joystick, and autonomous control. The design is optimized for low cost, since it uses a single sensor. The results demonstrate that the proposed algorithms are promising. They can also be used in various applications including UAVs, USVs, and smart robots. Future work will consider improving the navigation, guidance, and control modules, as well as using additional sensors to improve accuracy.

Reference

[1] R. Siegwart and I.R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*, The MIT Press, Cambridge, Massachusetts, London, England 2004.

[2] J. Borenstein, H.R. Everett, L. Feng, and D. Wehe, "Mobile Robot Positioning-Sensors and Techniques," ", *Journal of Robotic Systems, Special Issue on Mobile Robots*," Vol. 14 No. 4, pp. 231 – 249.

[3] S. Wong and Jennifer Yu, "Autonomous Stable Flight with a PID Controller,", Cornell University, Department of Computer Science, CS 4758/ 6758: Robot Learning Projects, Spring 2010.

[4] M.E. Holden,"Low-Cost Autonomous Vehicles Using Just GPS," ", *American Society for Engineering Education Annual Conference & Exposition*" 2004.

[5] Y.Zhao, GPS/IMU Integrated System for Land Vehicle Navigation based on MEMS, Licentiate thesis, Royal Institute of Technology (KTH), Sweden, September 2011,.

[6] J.W. Han, J.H. Park, J. Lee, S.K.Hong, "A PC Controlled GPS Guided Autonomous Ground Vehicle", *ICMIT* 2009 Poster.

[7] Satoshi Suzuki, "Autonomous Navigation, Guidance and Control of Small 4-Wheel Electric Vehicle," ",*Journal of Asian Electric Vehicles*," Volume 10, Number 1, June 2012.

[8] R.S. Christiansen, Design of an autopilot for small unmanned aerial vehicles, Master of Science, Department of Electrical and Computer Engineering, Brigham Young University, August 2004.

[9] "LPC2378 User Manual.", http://www.keil.com/dd/docs/datashts/philips/lpc23xx_um.pdf.

[10] "MCB 2300 Evaluation Board.", http://www.keil.com/mcb2300/mcb2370.asp.

[11] R.S. Burns, *Advanced control engineering*, first edition, Butterworth-Heinemann, November, 2001.

[12] http://en.wikipedia.org/wiki/Checksum.

[13] Y. Hong, J. Fang, and Y. Tao, "Ground Control Station Development for Autonomous UAV," ",*Proceedings of the First International Conference, ICIRA 2008*," Wuhan, China, October 15-17, 2008, Volume 5315, 2008, pp 36-44.