Low Cost Underwater Robot Sensor Suite

Matthew Joordens, *Member IEEE,* Jared Serna, Scott Songer, Casey Friday, Julie Hoy, Ralph Seiger,
Walter Madalinski *and* Mo Jamshidi, *Fellow IEEE*
Autonomous Control Engineering (ACE) Center and ECE Department
The University of Texas
San Antonio, TX, USA
matthew.joordens@utsa.edu

***Abstract*—One of the most expensive parts of underwater robotics is the sensors. This paper looks at modifying off the shelf components to create a sensor suite on a small budget. A big saving is made with sonar using a cheap commercial product to create a four sonar array. A depth sensor and acceleration navigation system are also developed.**

***Index Terms*—Sonar distance measurement, Sonar transducers, Transducers, Acceleration measurement**

I. Introduction

When designing autonomous underwater systems one of the more important aspects is the sensor suite. When one is working on a tight budget it can be crippling. A simple echo sonar unit alone can be from USD$2000 upwards. Here we look at some cheaper options.

The most expensive is the sonar unit. There is however a commercial unit used by fishermen to find fish that retails at under USD$30. This unit, the SmartCast made by Humminbird, can be modified to create an echo sonar unit with a range of 30m.

A simple search can produce low cost pressure sensors to determine the depth of the robot.

Navigation can be an issue. Ordinary GPS will not work underwater and underwater sonar, (a GPS on the surface with underwater sonar location) is expensive and limits the range of the robot. A dead reckoning system using accelerometers can be simply designed and the robot can surface for a GPS fix when errors get to large.

These three systems allow a robot to know its location, to navigate to another location and to perform obstacle avoidance. The systems here have been designed to work with a robot as described in Joordens, et al.[1]

II. Sonar

The Autonomous Control Engineering (ACE) of the University of Texas, San Antonio (UTSA) is dedicated to designing swarms of land, air and sea robots in a System of Systems (SOS).

This work was supported in part by University of Texas, San Antonio, Autonomous Control Engineering and by Deakin University.

M. A. Joordens is with the School of Engineering and Technology, Deakin University, Geelong, 3217, Australia (phone: +61-3-52272824; fax: +61-3-52272167; e-mail: matthew.joordens@deakin.edu.au).

To cater to the sea system the ACE lab has acquired a few underwater robots that could be remote controlled, and is now beginning the process of designing and creating the software and hardware that will transform the human-controlled underwater robots into autonomous robots. In order to work on a echo sonar unit for collision avoidance several Humminbird SmartCast remote sonar sensors, which are normally used by fisherman to determine the depth of the water and if there are any fish in the area, where aquired. The goal is to use these sonar sensors to keep track of the surroundings of the robot as it maneuvers underwater. In other words, we want to use the sonar devices as the "eyes" of the underwater robots. In order to do this, the lab had to reverse engineer the inner workings of the sonar devices. This was done by opening up the sonar sensor, analyzing its circuit, connecting the pins of the sonar sensor's microprocessor to an oscilloscope, reading the patent for the sonar sensor, and interpreting the information gathered.

Between reading the patent, and observing the signals of the sonar receiver and the radio frequency transmitter of the sonar device on the oscilloscope, certain conclusions were made.

It was concluded that the sonar sensor gather information by sending out pulses of sound and receiving the reflected echoes of these sound waves back through a piezoelectric crystal (quartz) transducer. The transducer converts the sound waves that are reflected back to the sonar device into analog voltage signals. These voltage signals are then converted into digital voltage signals. The receiving microprocessor of the SmartCast sonar system uses the time that it takes for the sonar pulses to be reflected back to determine the distance between the sonar device and the object that the sonar pulses are being reflected off of. The width, or strength, of the pulses are also used by the microprocessor to determine the relative size of the object that the sonar pulses are reflecting off of. For example, the longest pulse width will indicate that the object is the seabed, whereas smaller pulse widths could indicate that the object is a small fish (Fig. 1).

The top trace is the signal on the sonar transducer. The lower trace is the sonar units output. The first pulse is a synchronization pulse. The following pulses are acoustic echo returns. The time between the synchronization pulse and the following pulses is the time of a round trip of an acoustic wave from the sonar unit to the object and back. The width of the pulse equates to the magnitude of the return echo. It is assumes then, that the largest width represents the sea bed if it

978-1-4244-2173-2/08/$25.00 ©2008 IEEE

is in range. Other returns are either fish, other robots or are caused by spurious reflections.

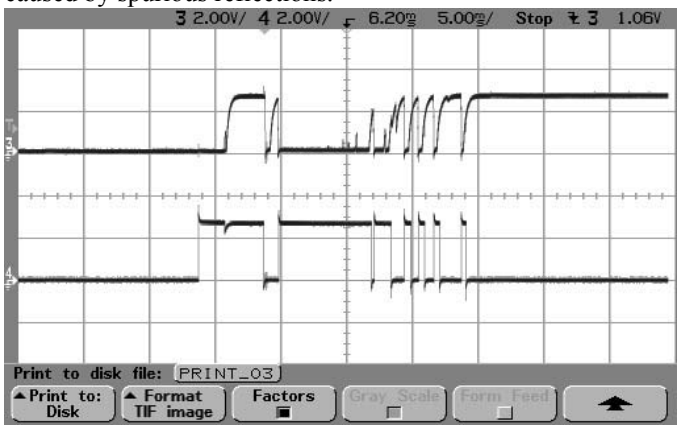

Fig. 1 Sonar unit output

After deciphering the meaning of the signals being transmitted out of the sonar sensor, we could now begin writing software code that would allow us to interpret these signals as distance measurements. we began first and foremost by outlining a list of requirements and possibilities for using one of the microcontroller boards (designed inhouse) to analyze the signal produced by the sonar unit that is sent to the radio frequency transmitter. A flow chart was then constructed based off of the idea of using different interrupt functions built into in house microcontroller. Based on this C code was then written to be programmed into the microcontroller. This code was written to keep track of when the sonar device was receiving sonar pulses as echoes, the time that was taken for these pulses to be received, the width of the received sonar pulses, and which had the longest pulse width. The software code was also written to analyze the information gathered and provide a distance measurement for every cycle of sonar pulses that are sent by the sonar device.

The distance measurements gathered by the sonar sensor, and interpreted by the microprocessor, are then sent through a serial connection to a computer and displayed on screen by the HyperTerminal program that normally comes standard with Windows' operating systems. However, at first, the numbers being displayed did not mean anything because the software code that was written had not yet been calibrated to relate the numbers being produced to actual distance measurements. In order to calibrate these numbers it was necessary to go to a swimming pool where we could conduct experiments that would allow us to compare the numbers being displayed on the computer to the actual depths of the swimming pool.

Testing the software and the hardware in the ACE laboratory before actually going to the swimming pool became essential since we only had one opportunity every week to conduct experiments. These experiments were conducted in a large wheeled garbage bin full of water. Changes or adjustments were made to the hardware and software after each experiment in preparation for the next.

Our experiments consisted of taking the sonar device, the microcontroller, a lap-top, and an oscilloscope to the swimming pool and examining the output of the sonar device through the hyper-terminal of the lap-top. We also would examine the output signal using the oscilloscope to ensure that the data that was being displayed on the lap-top was valid, and to take screenshots of the signal at different water depths. Without a doubt, every week posed new challenges to be resolved. However, after several weeks of testing our software, hardware, and gathering information on how it appeared that the numbers we were receiving corresponded to actual depths, we were finally able to begin the process of installing the sonar devices onto the underwater robots.

In order to install the sonar sensors onto the underwater robots, the lab acquired t a new set of SmartCast Humminbird sonar sensors, opened up four sonar sensors so that the appropriate wires could be soldered onto the circuits of the sonar sensors. The sonar devices were then connected to the microcontrollers inside of one of the underwater robots. Once connected, the sonar sensors were then strategically placed inside of the underwater robot in such a way that sonar readings could be taken from the front, bottom, and sides of the robot. On the first in pool experiment, we were able to record the information that the sonar sensors were gathering at the four different positions in which they were placed.

Each sonar unit was fired in turn. Whilst firing all at once was an option,[2] it simplified matters at this stage to treat each unit separately.

Based off of this information, we adjusted the software code in such a way that the underwater robot would remain at a certain distance away from the bottom of the swimming pool. The second experiment was to a good extent a success since the robot remained at a specific distance away from the bottom of the swimming pool. However, for some reason the robot would not stop turning to the right. Apparently there was an error with the readings of the sonar sensor that was placed on the right side of the underwater robot. This error was determined to be electronic noise, and was fixed with some noise suppression capacitors. The robots now have simple collision avoidance capability

Having four sonar units facing in different directions presents the option of being able to create a map of the environment and being able to navigate from that map.[3] This option may be used as a backup to accelerometer base navigation discussed later.

III. DEPTH

This section looks at the development of a depth sensor using an absolute pressure transducer. Added criteria for this was to interface and program a PIC microprocessor board to interpret the DC output voltage of the pressure transducer and relay the information as a control system for the depth of an underwater robot.

A. Design

After initial research about water pressure and pressure sensors, the first step was to find a low-cost, absolute pressure transducer that could go to depths as low as 50 ft. After some research, the Honeywell 19c100pa4k pressure transducer was selected and two were acquired, one for each robot.

The transducer, was mounted into an airtight casing to retrieve experimental data for its range of output voltage at different water depths. With a DC input voltage of 12 volts, the transducer was found to have an output of 22 millivolts at ground level and 29 millivolts 9ft underwater. This information was necessary to design a voltage reference circuit to scale the output voltage to 100 millivolts, which would be more suitable for interpretation in the microprocessor. A simple voltage divider was added to the 3.3 volts voltage reference already built into the board.**Fig. 2**

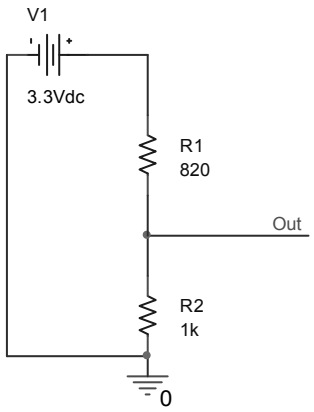

Fig. 2 Schematic of Voltage Reference Circuit

After implementing the voltage reference and wiring up the board with the transducer, code was written to display a hexadecimal output number corresponding to a given voltage output from the transducer and through the reference. It was found that the output was rather erratic and would need to somehow be stabilized. The first solution implemented was to add code to average every 10 hexadecimal output numbers by constantly filling a 10-element array and averaging the elements as they are received. This change in the code greatly stabilized the output number, but it was found that the DC input voltage from the battery power supply in the actual robot was much less constant than the DC power supply in the lab, which previous experiments were based on. The voltage from the battery was measured to range between 12.5 and 13.3 volts, constantly fluctuating and decreasing as the battery power drained with use. The next circuit to be added to the board was an 11 volt zener diode, which would cap the DC voltage from the battery at 11 volts. Fig. 3

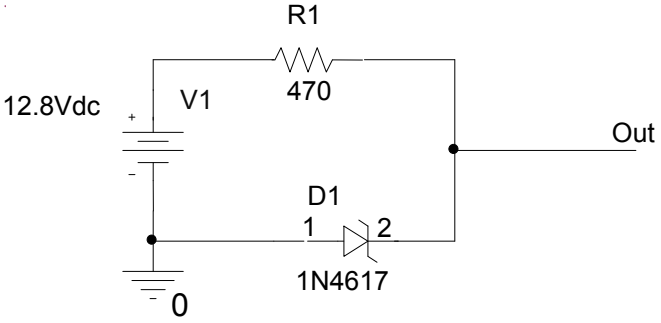

Fig. 3 Schematic of Zener Diode Circuit

The output of this circuit was then wired to power the pressure transducer (rather than having the battery wired directly to the transducer through the PIC board), thus creating a constant input voltage, and finally a more constant hexadecimal output number on the screen to represent the depth.

B. Testing

Now, all that was left to do was mount the transducer onto the robot, which was accomplished by drilling a hole and mounting the sensor to the wall of the robot's frame adjacent to where the board was to be mounted on the internal rack of processor boards. (Fig. 5) The code from the prototype board was burned onto the processor of the main board within the robot and the voltage reference and zener diode circuits added as well. Since the code for the depth was not too cumbersome, it was decided to integrate it on the robot's main processor in order to save space within the already crowded interior. (Fig. 4) The final experiment was successful and the hexadecimal output values were recorded as 22 at the surface, 27 at 5 feet, and 3C at 9 feet (the bottom of the pool). These numbers will be used in the future to program the robot to maintain a given depth, whatever the application may be. Accomplishing this is simple: if the depth output exceeds a given value (too deep), the robot's propellers will cycle on to raise its depth, and if the depth output drops below the given number, the propellers will cycle off until it falls too deep. The robot is already physically designed to be nearly neutrally buoyant, so with this new depth sensor, a given depth can be easily maintained. Furthermore, it will also be possible in the future for the depth output to be used in conjunction with the sonar to create a control system which will enable the robot to remain above a given maximum depth and also adjust for a changing floor as detected by the sonar. For example, the robot will be able to maintain a given depth within the deep end of a pool, but once it travels to more shallow waters the sonar will command the robot to rise to avoid collision with the floor, and maintain a given distance from the floor.

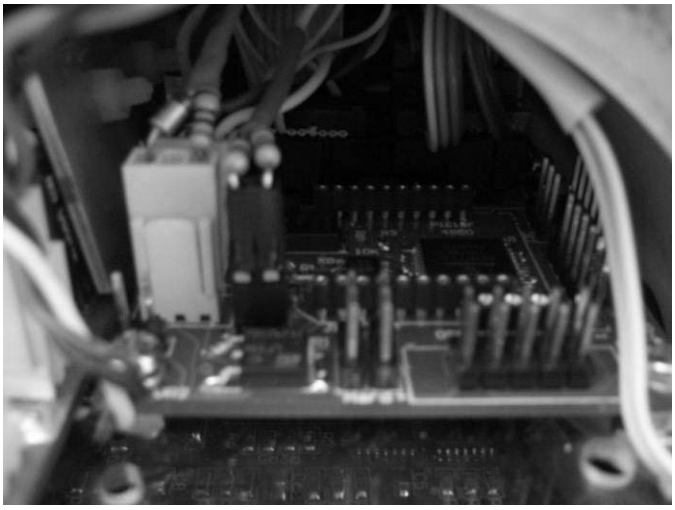

Fig. 4 Photograph of Microprocessor Board. Zener Diode (left) and Voltage Reference (right) Circuits Visible.

Fig. 5 Photograph of Mounted Pressure Transducer

IV. Accelerometers

The end goal of this underwater system was to create a swarm of robots, which rely solely on each other to perform tasks. Knowing each individual robot's position is crucial for mission accomplishment. Thus, the research was aimed towards using gyroscopes and accelerometers to acquire raw data, which would then be converted to useable data. The conversion involved forcing a system upon the devices and examining their output. Once this process was repeated enough to get quality test data, these data were put through rigorous scrutiny to obtain a venerable value of conversion between robot data and human interface data.

The use of accelerometers and gyroscopes is ideal in this setting, due to the following factors. The main competitor for tracking location is a Global Positioning System (GPS), which does not work under water. This constitutes the logical reasoning for the implementation of tracking with acceleration, also known as an Inertial Measurement Unit (IMU).

A. Data Translation

The first task involved in communicating with the robots was the translation of the accelerometer/gyroscope output. The gyro units are single axis and obtained from radio controlled systems. They accept a Pulse Width Modulated (PWM) signal for a servo motor and lengthen or shorten the width of the pulse based on the angular acceleration they experience. Three gyros are paced at 90 degrees to each other to capture the roll, pitch and yaw of the robot. A microcontroller is programmed to provide a signal of 20ms period with a 1.5ms pulse width. The returned signal is sampled to measure the new width.

The ADXL330 3 axis accelerometer was acquired to measure the acceleration in the 3 dimensions. It returns a 0 to 3.3volt analogue signal for each axis based upon the acceleration sensed. The microcontroller used has several analogue inputs that can connect directly to the ADXL330.

These accelerometers are piezo crystal based,[4] but does not use a single proof-mass.[5] The accuracy required does not need to go this expense.

B. Accelerometer Scaling

The acceleration value previously mentioned is not sent back in a form that can be used for manipulation, so it must be transformed into a unit suitable for operation, such as the metric system. Whilst the data sheet gives the acceleration in millivolts per G, to ensure that we have designed the interface and code correctly obtain the acceleration. A pulley system is used to perform these calculations. The accelerometer is attached to the microcontroller, which was attached to one end of the string. The other end held a weight, to pull the pulley down the track. Allowing the weight to fall would create acceleration on the horizontal axis. This caused an output from the accelerometer which should have been equal to approximately 9.8 m/s^2. Once a scaling factor is set between the accelerometer output and metric values, each output from the accelerometer is able to be scaled into a metric system value. It should now be possible to use the accelerometer output to calculate position.

C. Gyroscope Scaling

The scale factor of the gyro units was unknown. To find the acceleration scaling factor for the gyroscope, the gyroscope would be attached to the microcontroller and set on a spindle apparatus. A string is wrapped around the base of the apparatus, and the weight is released. As the weight pulls the string at the force of gravity, the string will spin the gyroscope in a rotational motion. This angular acceleration is also taken to be 9.8 m/s^2. A scaling factor would then be applied to the gyroscope output to cause each gyroscope output to be automatically converted to a metric value. This allows tracking of the total rotational motion of the gyroscope. This not only allows storing of the rotational motion, but aids in calculation of movement in two dimensions.

D. The Algorithm

After conversion from the output of the gyroscope and accelerometer to comprehensible values, acquisition of distance is necessary. An algorithm was used to convert

acceleration to position. This algorithm needed to be complex enough to have a low source of error but simple enough to implement easily in code. The chosen algorithm examined distance due to acceleration in each quadrant of the Cartesian coordinate system to provide accurate results. Once the algorithm was completed, it was tested with random test inputs to assure efficiency.

As the robot was designed to remain level it was decided not to use the angular accelerations for pitch and roll. This simplified the calculations. Using the accelerations for the horizontal x and y axis and the yaw and multiplying them by the change in time squared, ones get the horizontal distances and the angle. Simple trigonometry can then be used to determine to robot's new location.

The vertical z axis is treated separately. It is also multiplied by the change in time squared so that a vertical distance is obtained and used to determine the robot's depth.

E. Possible Sources of Error

The algorithm designed for this problem assumed that the robot's chassis is parallel to the horizon at all times. Tidal surges in the water could prevent this from being constantly true. This error could be corrected with additions to the algorithm that takes three axes of rotation into consideration.

The accelerometers themselves may introduce errors.[6, 7] These errors can only be fixed with a re-initialization of the system.

F. Translating Algorithm to Code

The Integrated Development Environment responsible for programming the Programmable Interface Controller uses the C programming language (Listing 1.). There were many factors to take into consideration when writing this code, such as what type of values should be stored as acceleration and distance. To use the math library correctly in code, all values were stored as floats. Using these values ensures the highest precision of the processors calculations.

```
  //get thetas from angular acceleration
  ztheta = zAngAccel * time * time;
  while(ztheta<0)  ztheta+=360;
    ztottheta += ztheta;
    while(ztottheta>360)
      ytottheta-=360;
    while(ztottheta<0)
      ytottheta+=360;
//get distance from acceleration
  xdist = xaccel*time*time;
ydist = yaccel*time*time;

//y plane assumed to be forward/backward
//x plane assumed to be left/right
if((ztottheta > 0 && ztottheta <= 45)||(ztottheta >
135 && ztottheta <= 225)||(ztottheta > 315 &&
ztottheta < 360))
{
  xtotdist += xdist*(cos(ztottheta));
  ytotdist += ydist*(cos(ztottheta));
}
else if((ztottheta > 45 && ztottheta <=
135)||(ztottheta > 225 && ztottheta <= 315))
{
  xtotdist += ydist*(sin(ztottheta));
  ytotdist -= xdist*(sin(ztottheta));
}
else if(ztottheta = 0)
{
  xtotdist += xdist;
  ytotdist += ydist;
}
else if(ztottheta = 90)
{
  xtotdist += ydist;
  ytotdist -= xdist;
}
else if(ztottheta = 180)
{
  xtotdist -= xdist;
  ytotdist -= ydist;
}
else if(ztottheta = 270)
{
  xtotdist -= ydist;
  ytotdist += xdist;
}
else
{
  xtotdist += xdist;
  ytotdist += ydist;
}
// Determine depth
zdist = zaccel*time*time;
ztotdist += zdist;
```

Listing 1. C code for algorithm

G. Communication in Code

The robots originally needed to communicate with a controller (user), and after final programming they will need to communicate with each other. To make this happen, the code implemented a packet system, where information was sent back and forth from the robot and a control system. Each packet consisted of 10 bytes. The first byte was an address byte, and the second was a size byte, describing how many remaining bytes would be used to send information. The remaining bytes send the total distance moved in each direction as well as the rotational position (yaw) of the robot.

H. Processing Data

Once the conversions and communications are complete, the output data is either sent to a user interface or stored in the robots' memory. Sending the data to a user interface would allow visible tracking of the robots and could allow a user to enter a destination point for the robot to travel to. The robots could also communicate autonomously, giving directions for a swarm to follow.

V. OTHER SENSORS

A. level

The robot that this sensor suite is designed for is able to control its pitch and roll. However to simplify operations it was decided to keep the robot level (or on an even keel). To do this a series of eight tilt switches were used. Four of the switches were set to detect a roll or pitch of more than 2 degrees from level and the other four were set at 5 degrees. Thus any significant roll or pitch can be easily countered.

B. Compass

There are various electronic compasses available at low cost. For instance DevanTech sell such a compass that presents its heading using an I^2C interface. It is accurate to within 4 degrees with a resolution of 0.1 degrees

C. .GPS

A GPS system cannot work underwater but is included the get a fix whenever the robot broaches the surface.

VI. Data collection

The sensor suite described is of necessity low level. It still has some overlap of data collection. The IMU/accelerometer system can be backed up by the depth gauge and the compass. Both the compass and the depth gauge are absolute measurements whereas the IMU is relative. Also the IMU's errors are accumulative. The confidence in the datum form a sensor will depend on these factors can be used to weight the data.

For absolute sensors the weighted data can be given as;

$$wa = da \times ka \tag{1}$$

Where: wa is the weighted data,
da is the senor's data, and
ka is the level of confidence (0 – 1).

For relative sensors the weighted data can be given as;

$$wr = dr \times kr \div t \tag{2}$$

Where: wr is the weighted data,
dr is the senor's data,
kr is the level of confidence (0 – 1), and
t is the time since data collection started

To fuse the data then;

$$w = \frac{\sum_{i=1}^{na} wa_i + \sum_{i=1}^{nr} wr_i}{Ka + Kr} \tag{3}$$

$$Ka = \sum_{i=1}^{na} ka_i \tag{4}$$

$$Kr = \sum_{i=1}^{nr} (kr_i \div t_i) \tag{5}$$

Where:
w is the weighted data
na is the number of absolute sensors
nr is the number of relative sensors
Ka is the sum of the absolute confidence
Kr is the sum of the relative confidence/time

The confidence in the relative sensors is divided by time as the more time the sensor has been collecting data the less accurate it is and the less confidence one has in it.

VII. Conclusion

The sensor suite described is now ready to be incorporated into several underwater robotic platforms. The low cost enables the budget to be stretched to equip a swarm of robots and can form the basis of the swarm data collection and control. The formula for data collection can be extended to gathering data from the other robots.

VIII. References

[1] M. Joordens, "Design of a low cost Underwater Robotic Research Platform," in *aper submitted to IEEE SoSE Confernec, Monterey, CA, USA*, Monterey, CA, USA, 2008, p. 6.

[2] S. Fazli and L. Kleeman, "A real time advanced sonar ring with simultaneous firing," *Intelligent Robots and Systems,* vol. 2, pp. 1872-1877, 2004.

[3] S. J. Lee, D. W. Cho, W. K. Chung, J. H. Lim, and C. U. Kang, "Feature based map building using sparse sonar data," *Intelligent Robots and Systems,* pp. 1648-1652, 2005.

[4] E. Cabruja, A. Collado, J. A. Plaza, and J. Esteve, "Piezoresistive Accelerometers for MCM-Package—Part II: The Packaging," *Microelectromechanical Systems, Journal of,* vol. 14, pp. 806-811, 2005.

[5] P. Cardou and J. Angeles, "Simplectic Architectures for True Multi-axial Accelerometers: A Novel Application of Parallel Robots," *Robotics and Automation,* pp. 181-186, 2007.

[6] M. F. Ying Kun Peng; Golnaraghi, "A vector-based gyro-free inertial navigation system by integrating existing accelerometer network in a passenger vehicle," *Position Location and Navigation Symposium,* pp. 234-242, 2004.

[7] F. A. Levinzon, "Noise of piezoelectric accelerometer with integral FET amplifier," *Sensors Journal, IEEE,* vol. 5, pp. 1235-1242, 2005.

Matthew A. Joordens (Member - IEEE, Fellow - The Institution of Engineers Australia) earned his Bachelor of Engineering (electronic) degree at Ballarat University in 1988 and a Masters of Engineering (by research) in Virtual Reality at Deakin University in 1996.

He began his career with Industrial Control Technology designing control systems to automate various different industrial processes. For 5 years he designed microprocessor based control systems for companies such as Ford, Pilkington Glass, Webtek and Blue Circle Southern Cement. He then moved to Deakin University and wrote their first electronics units. Using his industrial experience he designed one of the first Australian Engineering degrees in Mechatronics that still runs at Deakin as Mechatronics and Robotics. He currently lectures units in Digital electronics, Microcontrollers, Robotics and Artificial Intelligence after 15 years at Deakin. He is currently researching underwater swarm robotics in the USA.

Mr. Joordens is a Fellow of the Institution of Engineers, Australia and an IEEE member.