Research Journal of Applied Sciences, Engineering and Technology 6(5): 754-760, 2013
DOI: 10.19026/rjaset.6.4115
ISSN: 2040-7459; e-ISSN: 2040-7467
© 2013 Maxwell Scientific Publication Corp.
Submitted: March 02, 2012 Accepted: March 12, 2012 Published: June 25, 2013

Research Article

A Software Method for Generating Concurrent PWM Signal from Pic18f4520 for Biomimetic Robotic Fish Control

M.O. Afolayan, D.S. Yawas, C.O. Folayan and S.Y.Aku
Mechanical Engineering Department, Ahmadu Bello University, Zaria

Abstract: A method of generating multiple pulse width modulated signal with phase difference is presented in this work. The microcontroller used is PIC18F4520 and its output is used to drive Futaba RC servo motors directly. The concurrency of pulse width modulated signal in this work is relative, due to the fact that there is a finite time microcontroller period) between each instruction toggling the output pins. This finite time is equal to the minimum period of the microcontroller, which is 125 ns in this work. Also, the phase difference for the servo motors is set to 60_o and is achieved by changing the duty cycle of each of the channels while the period remains at 20 ms for all the channels. The robotic fish using this software PWM code was able to attain a linear speed of 0.985 m/s.

Keywords: Biomimetic, concurrent PWM, fish robot, PIC microcontroller, software PWM

INTRODUCTION

"Pulse Width Modulation (PWM) modules are commonly used in many applications to provide an inexpensive control output method that uses only a few external components". Typical use in embedded control applications light dimming, motor speed control, output voltage control, and communication between devices, drive switches in a power conversion. Or, it can be filtered using external components to produce an averaged 'analog' signal with an output level that is proportional to the duty cycle (Lucio, 2006; Parthiy, 2007). Information is commonly coded as the duty cycle of the PWM signal while the frequency (or period) remains fixed. The duty circle can vary from 0 to 100% as shown in Fig. 1.

Many PIC® microcontroller (MCU) comes with built in PWM generation capability (hardware PWM) (Fig. 2 shows its schematic). For PIC18F4520 microcontroller, the PWM share its hardware design with CCP (Capture/Compare) module and ECCP (Enhanced Capture/Compare) module. Each module contains a 16 bit register which can operate as a 16-bit Capture register, a 16-bit Compare register or a PWM Master/Slave Duty Cycle register, the PWM has 10 bit resolution (Microchip Technology Inc, 2008). However several situation do arise that the user want more PWM output pins and had to create it in code (software PWM). This article focuses on software techniques used for generating PWM signal and in particular a method is described here that was actually used as a building block of a hyper-redundant robot in the form of a fish.

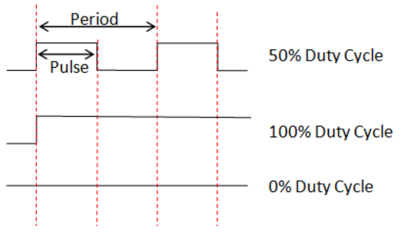

Fig. 1: PWM output showing the period and the duty circle

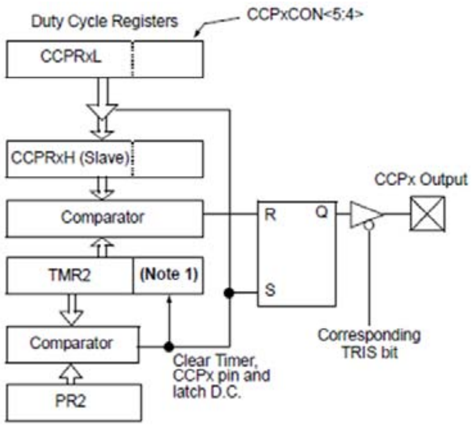

Fig. 2: A typical microchip MCU hardware pulse–width modulation module (Microchip, 2008)

The problem: Robots such as hyper-redundant robots require several actuators to be managed simultaneously. The test robot (Fig. 3) used for this work also need to generate ultrasonic signals, one for distant measurement

Corresponding Author: M.O. Afolayan, Mechanical Engineering Department, Ahmadu Bello University, Zaria

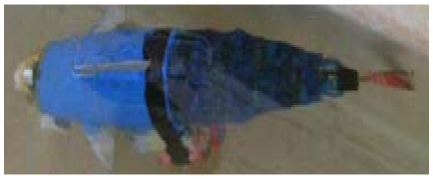

Fig. 3: The biomimetic fish robot for which the PWM code was developed for.

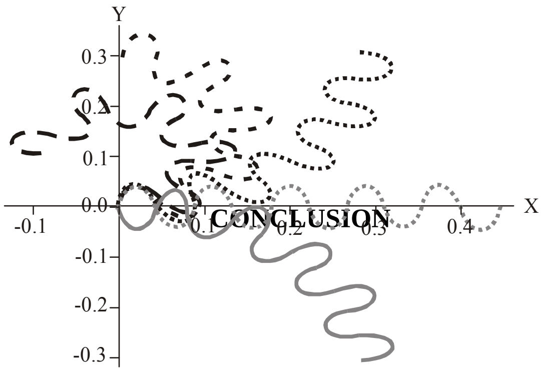

Fig. 4: Serpenoid curves (Kevin, 1997)

and another for object detections and avoidance lgorithm while using just only one microcontroller. The PIC® microcontroller used (PIC18F4520) comes with 2 hardware PWM and obviously will not be sufficient for the robot. The solution then is to go for software PWM.

Common software methods for generating software PWM signal: There are 3 methods commonly used by programmers according to Ole (1997), they are:

- Use of the instruction cycle as a time base for pulse generation
- Use of dynamic adjustment of the program counter
- Use of timed interrupt

Using the instruction cycle as a time base for pulse generation: In this method, the period of the PWM signal is based on counting the number of instruction cycle. For example if the instruction takes 1 μs to execute, it implies a count of 1000 instruction cycle will be needed to get a PWM with 1 ms pulse. Timer pre-scaling is used for achieving longer period. Generating PWM this way consume much of the microcontroller time. Also, the resolution is greatly dependent on the instruction cycle.

Using dynamic adjustment of the program counter: The design of microchip microcontroller is such that the Program Counter (PC) can be adjusted dynamically to alter the next instruction to fetch and execute. If a pin is set high by several similar instructions X_n and each lasting for T_{CY} (usually the instruction cycle period), then:

Pulse length = $(X_1 + X_2 + X_3 + \ldots X_{n-3} + X_{n-2} + X_{n-1} + X_n) * (T_{CY})$

To use this method, the X_1 or X_2 or X_3 etc is jumped (cumulatively) so that the final period is altered. Like the previous method, the resolution is also dependent on the instruction cycle, the higher the clock frequency, the better the resolution. Furthermore, much memory will be used if long period is desired. This method cannot handle multiple PWM signal concurrently.

Here is an example reproduced from Microchip application notes AN00654 [4] to buttress the point

```
.
PORTA equ 04h
LENGTH equ 0ch
.
movlw 4 ; value for pulse length of 1 Tcy
movwf LENGTH ;
.
movf LENGTH,W ; This is an indirectly addressed
relative jump
addwf PCL,F ;
bsf PORTA,3 ; start high pulse 5 Tcy
bsf PORTA,3 ; start high pulse 4 Tcy
bsf PORTA,3 ; start high pulse 3 Tcy
bsf PORTA,3 ; start high pulse 2 Tcy
bsf PORTA,3 ; start high pulse 1 Tcy
bcf PORTA,3 ; end high pulse
```

Using timed interrupt: This method tries to take away or reduce the time the microcontroller dedicates to generating the PWM signal. After the port is set, the timer is loaded with an adjusted number corresponding to the desired pulse length. After the timer overflows, the interrupt service routine starts and resets the port. This method can generate PWM signal with long period and duty cycle with less demand on the microcontroller time. The minimum resolution is still the instruction cycle period (for Microchip microcontroller it is ¼ of the system clock). Because of the use of interrupt, there is minimum time dedicated to servicing the interrupt routine (the overhead) and therefore, the PWM period cannot be less than it.

Motion generation in hyper-redundant robots: There are three approaches (Kevin, 1997) used in controlling hyper-redundant robot joints, they are:

- Serpenoid curve method
- Follow the leader approach
- Built in motion pattern

The Serpenoid curve (Fig. 4) was a result of realization of study on live snakes by Hirose in Japan (Yamada and Hirose, 2006) from which he discovered that snake motion does not follow sine wave as thought earlier. The equation for the Serpenoid curve derived is shown in Eq. (1), snake like motion does not follow sine wave as thought earlier. The x(s), y(s) represent the displacement in the x and y directions respectively measured along the curve body length s, l is the body length and J(α) are Bessel functions, m are joint osition. A plot of this parametric equation will yield Fig. 4 curves:

$$x(s) = sJ_o(\alpha) + \frac{4l}{\pi}\sum_{m=1}\frac{(-1)^m}{2m}J_{2m}(\alpha)\sin\left(m\pi\frac{s}{l}\right)$$

$$y(s) = \frac{4l}{\pi}\sum_{m=1}^{\infty}(-1)^{m-1}\frac{J_{2m-1}(\alpha)}{2m-1}\sin\left(\frac{2m-1}{2}\pi\frac{s}{l}\right) \quad (1)$$

In follow the leader approach, the head (or the tail if reversing) segment is controlled. The information is passed to the next segment till the last segment is reached. The desired motion is mathematically generated. The use of built in motion pattern means that the microcontroller (or microprocessor) will have to adjust each segment according to the predefined motion stored in its memory. This is less mathematically involving but not very flexible.

For a small robot, the power demand and size demand implies that battery be used and hence microcontroller based computing system is the first option in controller selection. For the same reason many researchers will prefer the last two methods of motion generation as floating point mathematics are avoided.

Objective of this work: The requirement of the robotic fish tail for swimming (and many other hype redundant robots) is that the actuators operate at some phase angle to each other in other to generate motion. There are researchers that have used multiple microcontrollers for each actuator/motor (Jindong and Huosheng, 2003; Jindong and Huosheng, 2004; Kato, 2000; Liang, 2002; Yu *et al*., 2002) but in this work, it is intended to use a single microcontroller to perform all the peripheral functions and especially the PWM signal generation-all concurrently while still handling the phase shifting.

A METHODOLOGY FOR GENERATING CONCURRENT PWM SIGNAL

For this particular project, the specifications for the PWM signal needed are:

- Three rigidly coupled PWM signal that is out of phase by 60º
- Continuously varying duty cycle
- Different duty cycles at any point in time

Fig. 5: Futaba S3003 RC servomotor

- The same period of 20 ms
- Repeated (introduced dead band)
- Less load on the processor time

Development environment: Microchip MPLAB IDE v8.56 in Microsoft Windows 7 (64 bit) environment was used for the work. MPLAB SIM simulator was used for all software simulation and Microchip PICKit 2 for hardware simulation. The Microcontroller is PIC18F4520 running at 32 MHz (simulated and on the real chip). Futaba S3003 RC servomotor Fig. 5, was used to test the PWM output directly.

Description of the concurrent PWM signal generator: The Concurrent PWM signal generator is a modified timer based interrupt method with much contribution from instruction time method. Timer0 interrupt (INT0) was used as the trigger. The period is fixed at ≈20 ms. Flowchart of Fig. 6 shows the basic PWM model. The pulse length could be a table of pre calculated values (built in motion pattern) or range of value that could be picked from. It could also be mathematically generated values. The processes involved in generating the data for the pulse length are performed during the time between the pulses. As soon as there is an interrupt, critical register are saved and the Timer0 is prepared for the next interruption. The Ports are all set high, the difference (lag) between the start of the first and the last is calculated as follows:

If Port 0 start time = T_{port0}
Then Port X start time = $T_{port0} + T_{port1} + T_{port2} + \ldots. + T_{port\ x-1} = T_{port0} + (x-1) * T_{CY}$

where T_{CY} is the length of an Instruction Cycle
The PIC microcontroller used was run at 32 Mhz and

therefore the T_{CY} = (1/32 Mhz)*4 = 0.000000125s or 125ns lag for Port x = (x-1)* T_{CY} = (x-1) * 125ns

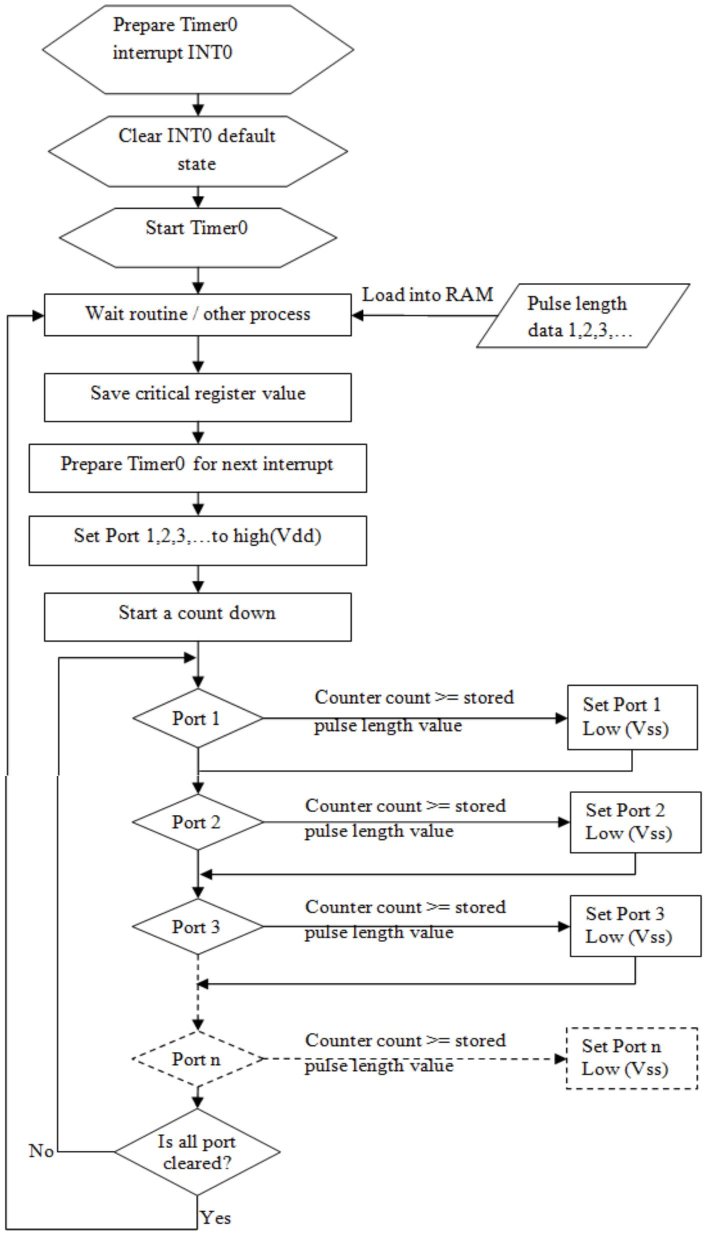
Prepare Timer0 interrupt INT0
Clear INT0 default state
Start Timer0
Wait routine / other process
Load into RAM
Pulse length data 1,2,3,…
Save critical register value
Prepare Timer0 for next interrupt
Set Port 1,2,3,…to high(Vdd)
Start a count down
Port 1
Counter count >= stored pulse length value
Set Port 1 Low (Vss)
Port 2
Counter count >= stored pulse length value
Set Port 2 Low (Vss)
Port 3
Counter count >= stored pulse length value
Set Port 3 Low (Vss)
Port n
Counter count >= stored pulse length value
Set Port n Low (Vss)
Is all port cleared?
No
Yes

Fig. 6: Flow chart of the basic PWM generator

As an example, the lag between the first port and last port of a 6 PWM concurrent generator will be:

$$\text{Lag} = (6-1) * 125\text{ns} = 625\text{ns}$$

Fortunately, this lag will be fixed as the PWM signals have the same reference, i.e., they have the same period that is fired up by a single timer (Timer0). The period for any channel will therefore remain constant at ≈20 ms as designed. Using Fig. 8:

$$T_1 = T_2 = T_3 \approx 20 \text{ ms.}$$

The count down process check and compared the RAM value of the pulse against the current count, if greater or equal, the corresponding port is set low. The routine is exited as soon as all the ports are set low.

A phase difference is needed for the robotic fish segment for motion to take place. These motors are made to rotate at different phase angle by adjusting their duty cycles at different rate. This calls for modification of the basic PWM code generation procedure. The modified flow chart with phase generator is as shown in Fig. 7. The phase data can also be a table of pre calculated values or mathematically generated values. The reference to when to change the motor phases is derived from the current state of the ports. The phase data modifies the pulse length data by simply controlling whether to load new values or not.

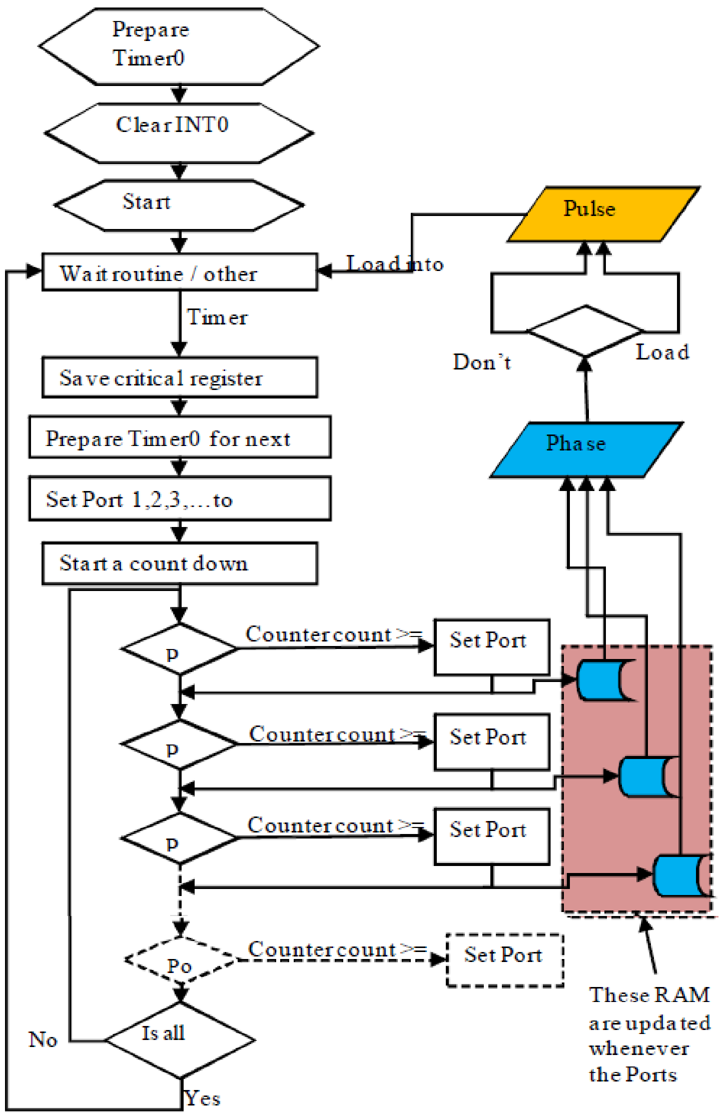

Fig. 7: Flow chart of the modified PWM generator

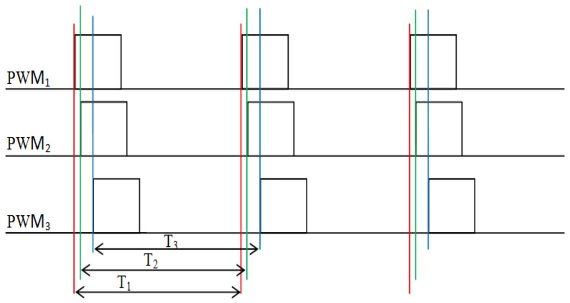

Fig. 8: An exaggerated illustration of lag present in the concurrent PWM generator

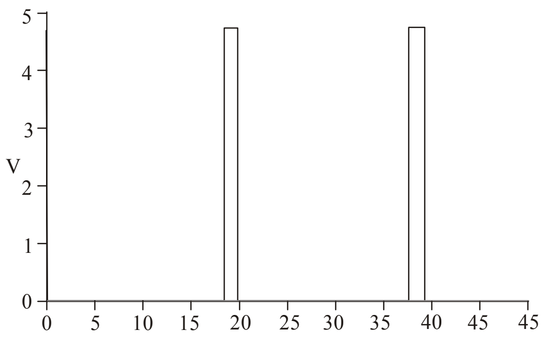

Fig. 9: Oscilloscope plot of one channel showing repeated pulses at ≈20 ms

This automatically creates a dead band. This dead band is used advantageously by the fact that the inertia of the servo motor shaft and gear cannot keep up with a rapidly changing PWM duty cycle. The phase controller is therefore actually causing the PWM signal generator to "wait" for the motor to finish up. These waiting periods is varied for different servo motor and thus lead to phase lags or leads.

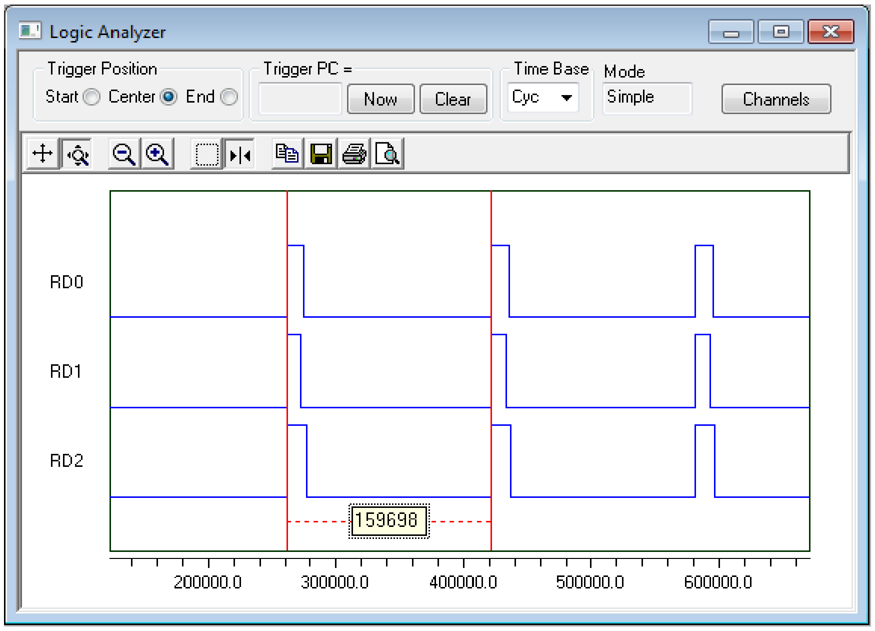

Fig. 10: Simulated PWM showing the varying duty cycles all occurring simultaneously at ≈20 ms interval. This image was a screen capture of the routine under test for three output

RESULTS AND DISCUSSION

Three tests were performed to assess the performance of the concurrent PWM generator, they are:

- Test to assess the accuracy of the period of the PWM signal generated
- Test on the concurrency of the output PWM signals
- Field test to confirm the phase differences actually leads to movement.

To test for the accuracy of the period of the PWM signal generated, an oscilloscope (Fig. 9) was used for measuring the voltage level (with reference to V_{ss} or the ground) of the output pins while the firmware was running at 32 Mhz. It can be deduced from Fig. 9 that the PWM period is 20 ms as designed.

To test for the concurrency of the output PWM signals, an MPLAB IDE v8.56 in built simulator Logic analyzer (Fig. 10) was used for capturing the port status of the microcontroller PWM output pins. The tests were performed without any servo motor connected and at a supply voltage (V_{dd}) of 5V. Figure 10 shows three WM signals starting “simultaneously” although with a difference of 125 ns between adjacent channels or port (RD0, RD1 and RD2). Also, each channel has different width, the information as shown translate to 60_o phase differences of the servo motor outputs.

The field test was performed with the fish robot inside stationary body of water. The field tests was done in a wooden water tank (60.96x121.92x60.96 cm) filled with water to a depth of 30 cm or pressure head of 2.91 kPa (Fig. 11). For the test, a digital camera-Sony Cyber-shot digital camera (model DSC-S730) set to VGA mode was used for video capture.

Figure 12 shows in graphical form the swimming speed with respect to different tail (peduncle) amplitudes at frequencies of 0.5, 0.7, 0.9, 1.1, 1.3, 1.5 and at the maximum (1.7 Hz obtained inside water).

Fig. 11: The robot inside the wooden water tank

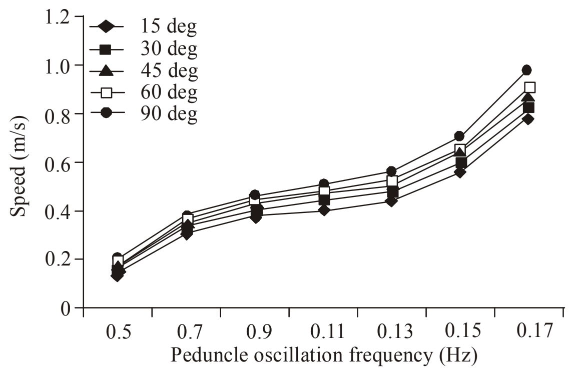

Fig. 12: Speed of robot against peduncle (last segment) oscillation at different peduncle amplitude

The maximum oscillation achieved was 1.7 Hz, measure with slowed video capture of the fish robot and maximum linear speed was 0.985 m/s. This maximum speed is one third (1/3) of that of a life Mackerel fish that was imitated in this design.

CONCLUSION

A software method for generating concurrent pulse width modulated signal for controlling a futaba 3003 servo motor for biomimetic fish robot application has being presented in this work. Also, the concurrency and phase difference works very well. A practical implementation of this control scheme made the robot to achieve a record 0.985 m/s or 1/3 of a life fish (mackerel that was imitated in design).

ACKNOWLEDGMENT

This project was supported by Macathur Foundation and Ahmadu Bello University Board of Research Grant.

REFERENCES

Jindong, L. and H. Huosheng, 2003. Proceedings of the 9th Chinese Automation & Computing Society Conference in the UK, Luton, England, 20 September 2003. Building a Simulation Environment for Optimising Control Parameters of an Autonomous Robotic Fish.

Jindong, L. and H. Huosheng, 2004. Building a 3D Simulator for Autonomous Navigation of Robotic Fishes. Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems. Semptember 28-October 2, 2004, Sendai, Japan.

Kato, N., 2000. Control performance in the horizontal plane of a fish robot with mechanical pectoral fins. IEEE J. Oceanic Eng., 25(1): 121-129, 0364-9059.

Kevin, J.D., 1997. Limbless locomotion: Learning to crawl with a snake robot. Ph.D. Thesis, At the Robotics Institute Carnegie Mellon University, 5000 Forbes Avenue, Pittsburg, PA, 15213.

Liang, J., 2002. Researchful development of underwater robofish ii- development of a small experimental robofish. Robot, 24(3): 234-238.

Lucio, D.J., 2006. A Technique to Increase the Frequency Resolution of PICmicro® MCU PWM Modules. Microchip Technology Inc. application note: AN1050

Microchip Technology Inc, 2008. PIC18F2420/2520/ 4420/4520 Data Sheet.

Parthiv P., 2007. Software PWM Generation for LED Dimming and RGB Color Applications.Microchip Technology Inc. application note: AN1074

Yamada, H. and S. Hirose, 2006. Development of practical 3-dimentinal active cord mechanism ACMR4. J. Robotics Mechatronics, 18(3): 305-311.

Yu, J.Z., E.K. Chen, S. Wang and M. Tan, 2002. Research Evolution and Analysis of Biomimetic Robot Fish. Contr. Theory Appl., VOL: pp.