Automatic Robotic Arm for Nucleic Acid Tests

Group1
Xiaoyang Sheng 519021910884
Zheyu Zhang 519370910094
Tiancheng He 519370910165

1. Background

Under the influence of Covid-19 epidemic, people's life has been greatly affected. During the toughest days, even food supply has been cut off. As time proceed, though the basic supplies have recovered, nucleic tests are still companying us all day.

I believe everyone has experienced lining up for a nucleic acid test. The biggest problem causing the waiting is the lack of trained medical workers. As we all know, human-resource is the most valuable resource today, especially those with special techniques.

As students majoring in ECE and as victims standing in the sun for hours, what can we do to alleviate the situation? Ideas come into our mind, "What if one worker can do the acid test for ten persons at the same time?"

What about a mechanical device doing the repeated works for us? That's when a nucleic acid test helper is needed. Our final version of nucleic acid testing helper is a Pic 32 driven robot that medical workers can remotely send instructions for both the mechanical arm and the patient.

Our final version of Nucleic Acid Testing Helper is a PIC-32 driven robot that achieves functions that medical staff can send the orders by remote control of Bluetooth to make the robotic arms accomplish the nucleic acid test without close contact.

2. Components

Fig 1. LCD.

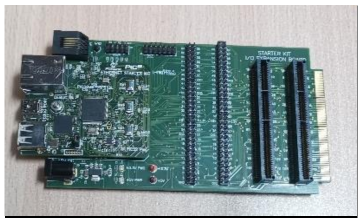
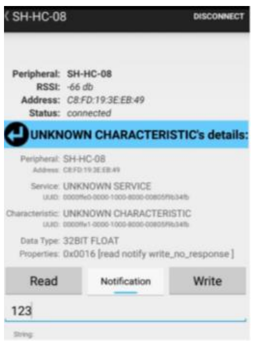


Fig 2. PIC32 MCU.

Fig 3. HM-10 blue tooth.


SH-HC-08

Peripheral: SH-HC-08
RSSI: -66 db
Address: C8:FD:19:3E:EB:49
Status: connected

UNKNOWN CHARACTERISTIC's details:

Peripheral: SH-HC-08
Address: C8:FD:19:3E:EB:49
Service: UNKNOWN SERVICE
UUID: 0000ffe0-0000-1000-8000-00805f9b34fb
Characteristic: UNKNOWN CHARACTERISTIC
UUID: 0000ffe1-0000-1000-8000-00805f9b34fb
Data Type: 32BIT FLOAT
Properties: 0x0016 [read notify write_no_response]

123

Fig 4. BLE Scanner.

Fig 5. MG996R servo.

* **LCD:** 2 lines. 16 characters per line.
* **PIC32 MCU:** The core control unit.
* **HM-10 blue tooth module:** UART protocol. Baud rate: 9600Hz. Maximum transmission distance: 20m.
* **BLE Scanner:** An android app used to connect one's cell phone to the HM-10 blue tooth module.
* **MG996R servo:** PWM period = 2ms. Rotation angle is proportional to duty cycle.

Duty cycle (ms)	Rotation angle (degree)
1.5	0 (Initial position)
2.5	-90 (Counter-clockwise)
0.5	90 (Clockwise)

3. Design Details

3.1 UART

We configured port *U1RX* to receive input strings transmitted from the blue tooth. The input strings have a maximum size of 32 characters, since the LCD is *2 lines x 16 characters per line*, and will be stored in a 32-character sting buffer. Interrupt will be triggered if UART receiver buffer is not empty.

3.1.1 Capture Input Strings

To properly capture input strings, we set a rule that all input strings should end with a ';'. In ISR, the program continuously reads the UART receiver buffer and stores the newly read character in the string buffer. If the string buffer is full, the character will be stored in the first slot of the buffer. It is the users' job to ensure each input string does not exceed 32 characters. If the character is a ';', then a complete input is captured.

3.1.2 Process Captured Strings

If the string is "exec", that is, robotic arm performing a nucleic acid test, the program set the CFORCE bit of the DMA channel 0, which starts to send sequences of duty cycles to the five servos, triggering a smooth movement of the robotic arm. If the string is something else, that is, some instructions like "Please open your mouth!", the program prints the string on the LCD.

3.2 PWM

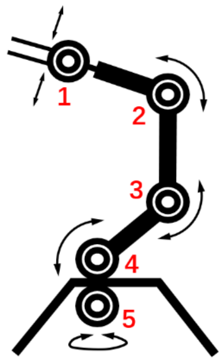


Fig 6. Robotic arm.

We used all five PWM modules to control the five servos respectively, as is shown in the figure. (Servo 1 controls the hold and release of the clamp.) The duty cycles are set by several DMA operations, which will be further elaborated in the DMA part.

Besides, since the 5V voltage supply of the PIC32 board for 5 servos is not power enough, we purchase an independent battery and one servo-driven power module to make the power supply stable and easy to manage.

3.3 DMA

3.3.1 Motivation

We found the rotation speed of the servos too fast, which may make the robotic arm tilt over. Therefore, to make servos rotate gently to the desired angles, we have to send a sequence of duty cycles at a proper interval. Here DMA is a good choice, which can automatically send data from a source address to a destination address, avoiding code redundancy and heavy CPU interference.

3.3.2 Basic Setup

We used all eight DMA channels. Performing a nucleic acid test can be broken down to three phases. Fist, servo 2, 3 and 4 rotate to lift up the robotic arm, which requires 3 DMA channels for each servo, channel 0, 1 and 2. Second, servo 2 and 5 rotate to make the clamp draw a circle, mimicking the sampling process in a recipient's mouth, which requires another 2 DMA channels, channel 3 and 4, for each servo. Third, servo 2, 3, 4 rotate to lay down the robotic arm, which requires the left 3 DMA channels, channel 5, 6 and 7, for each servo.

A helper function is written to generate the correct PWM duty cycles array by specifying starting degree, end degree and interval. The detailed configuration of 8 DMA channels and 6 DMA ISRs could be found in *DMA_init()* and *DMAx_ISR()* the source code.

3.3.3 Synchronization Between Channels

We carefully designed the synchronization strategy between all 8 DMA channels, taking advantage of cell-transfer-complete and block-transfer-complete interrupts.

In phase 1, servo 4 is the master while servo 2 and 3 are slaves. Servo 4's DMA channel interrupts at every cell transfer completion. In its ISR, the program sets the *CFORCE* bits of servo 3's and then servo 2's DMA channels to trigger a cell transfer. Each transfer is followed by a 10ms delay. Therefore, servo 2, 3 and 4 act like rotating simultaneously. Servo 3's DMA channel requires no interrupt. Servo 2's DMA channel interrupts at every block transfer completion, which means that when this channel interrupts, servo 2, 3 and 4 have all completed their transmission, and that the lift-up phase completes. In this ISR, the program set the *CFORCE* bit of servo 3's phase 2 DMA channel and enters phase 2.

Similarly, we configured cell-transfer-complete interrupt for servo 2's phase 2 channel and block-transfer-complete interrupt for servo 5's phase 2 channel, then cell-transfer-complete interrupt for servo 4's phase 3 channel and block-transfer-complete interrupt for servo 2's phase 3 channel.

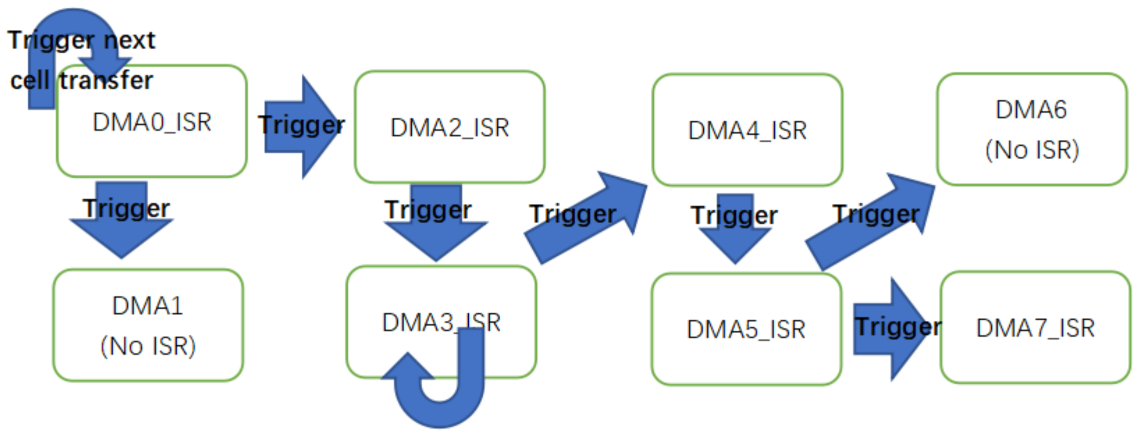


Fig 7. DMA ISR relation.

4. Demo

Welcome to watch and share our project demo at:
https://www.bilibili.com/video/BV1at4y1V7TA?share_source=copy_web&vd_source=23ea60629e0226ead06f47ebf302f91b

5. Future Plans

* Improve DMA channel synchronization mechanism.
* Implement more robotic arm movements
* Add multiple robotic arms.