# CG2271 Mini-Project Report

**Pham Quang Minh, A0170723L | Huu Nguyen, A0170745A | Putra Danish, A0164802J**

In this report, we will elaborate on the tasks that we have implemented for the project, as well as provide an in-depth explanation for the overall RTOS architecture.

## Tasks

For this project, we used a total of 7 tasks to be scheduled appropriately for the robot. The tasks are as follows, according to priority (Higher number corresponds to higher priority):

| Priority | Task | Task Code Name | Period (ms) |
|---|---|---|---|
| 6 | Bluetooth | blueToothTask() | 25 |
| 5 | Motor | motorTask() | 25 |
| 4 | Buzzer | buzzerTask() | 25 |
| 3 | Green LED | greenLedTask() | 25 \| 100 |
| 2 | Red LED | redLedTask() | 250 |
| 1 | Baby Shark Buzzer | babySharkMusicTask() | 400 |
| 1 | End Buzzer | endChallengeMusicTask() | 400 |

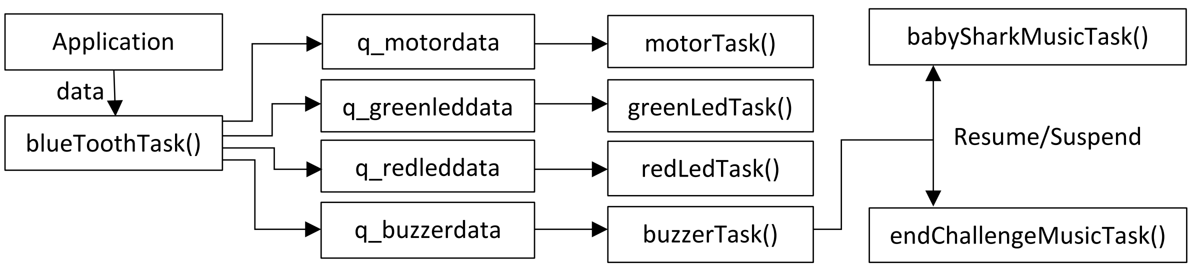
We schedule the tasks based on RMS priority scheduling policy and the importance of each task.

## Overall Architecture

In order to handle all the data inputs, we designed the architecture to have 4 queues. Each of the 4 queues handles data received from the Bluetooth Task and carries out the appropriate action, according to the task(s) it handles. The table below summarizes each queue.

| Queue Code Name | Data Handled | Task(s) Handled |
|---|---|---|
| q_motordata | Motor | motorTask() |
| q_greenleddata | Green LED | greenLedTask() |
| q_redleddata | Red LED | redLedTask() |
| q_buzzerdata | Buzzer | buzzerTask() |

A simplified diagram of the architecture is shown below.



## Bluetooth Task

The Bluetooth Task checks if Serial is available every 25ms, while the Smartphone controller application will only send new data when a button is pressed or released. If Serial is available, the Bluetooth task sends data input from the application to the appropriate queues. Each data input from the application comes in 1 byte. The data from the received byte can be the Bluetooth connection signal, the start/end challenge run signal, the stop signal or one of the eight direction that the robot is required to move.

**Motor Task**

The Motor Task handles the movement of the robot. It is set to blocked to wait until it can retrieve new data from the q_motordata queue. After receiving the data, the task makes calculations to determine the PWM duty cycle for each pair of motors on both sides depending on the direction it is requested to move. The task will then do a one-time analogWrite() to all required pins to control the movement of the robot before returning to the blocked state and repeat the cycle of waiting for new data. This allows the motor to run only when the button is held down on the Smartphone controller application, instead of continuously running with one press of the button.

**Green LED Task**

After initializing all 8 LEDs to turn on, the Green LED Task is blocked while waiting for the Bluetooth connection signal from the q_greenleddata queue. When it receives the signal, it will control the 8 LEDs to flash twice. Then, all 8 LEDs are set to on and the internal state is set to stationary. From then on, it will run periodically with a cycle of 100ms. At the start of each cycle, it will check for new data from the queue without waiting. If there is new data, it will dequeue the data and change the internal state accordingly. After that, it will control the output of the LEDs according to the current internal state by writing digital signals to a Shift Register. It will either only set one of the LEDs to turn on (which is determined by the LED running mode pattern) when the robot is moving or light up all 8 LEDs when the robot is stationary.

**Red LED Task**

The Red LED Task runs in cycles of 250ms. Similar to the Green LED Task, it will check new data from the q_redleddata queue at the start of the cycle and change the internal state accordingly. It will then decide how it would toggle its output signal to the 8 red LEDs depending on its current internal state. Toggle once every cycle when the robot is stationary and toggle once every two cycles when the robot is moving.

**Buzzer Task**

At the start, the Buzzer Task waits for the Bluetooth connection signal in the same way as the Green LED Task. When the signal comes, it will play the Bluetooth connection tone once. After that, it begins its cycle of waiting for new data (in blocked state). This is similar to the Motor Task. When new data arrives, it will dequeue from q_buzzerdata and determine if it should resume or suspend either the Baby Shark or the End Challenge Task. If the robot is beginning the challenge run, the Baby Shark Task will run or resume, while the End Challenge Music Task is suspended. Conversely, if the robot is ending the challenge run, the End Challenge Music Task will run or resume, and the Baby Shark Task is suspended. The instruction to run or resume is determined by the current state of the Task.

**Baby Shark & End Challenge Music Task**

The Baby Shark & End Buzzer Task utilizes the Arduino's Tone library to play different frequency tones with varying delay times to produce music that closely resembles the famous baby shark song and Harry Potter™ theme. At the start, both tasks are suspended until one of them is resumed by the Buzzer Task. Once a task starts, it will continuously play its piece of music in infinite loop until it is suspended by the Buzzer Task. At all times, there will be at least one of the two tasks suspended. The cycle of both tasks is one-eighth of the time duration of a basic music note duration (400ms).

**Conclusion**

One problem we faced at the end of the project is the 9V battery being unable to supply enough power to the system.

We were unable to understand what the exact problem was, but we have deduced it to be a hardware problem. Upon frequently changing the 9V battery, we have concluded that using an Energizer battery works better for the robot as compared to an EverReady or GP Ultra battery.

Moreover, some of the wires were faulty and it took us a few nudges to ensure a proper connection. In hindsight, we could have soldered the connections to ensure the connections were secure.

Despite this flaw, the robot met all the criteria for this mini-project and hence was a success.