SIMON FRASER UNIVERSITY

ENSC 351

PROJECT YIPPEE:

The Automatic Robot Alarm Clock

Qiraa Qadri - 301454940

Eugenia Demendeeva - 301461757

Sundus Subuktegin - 301472873

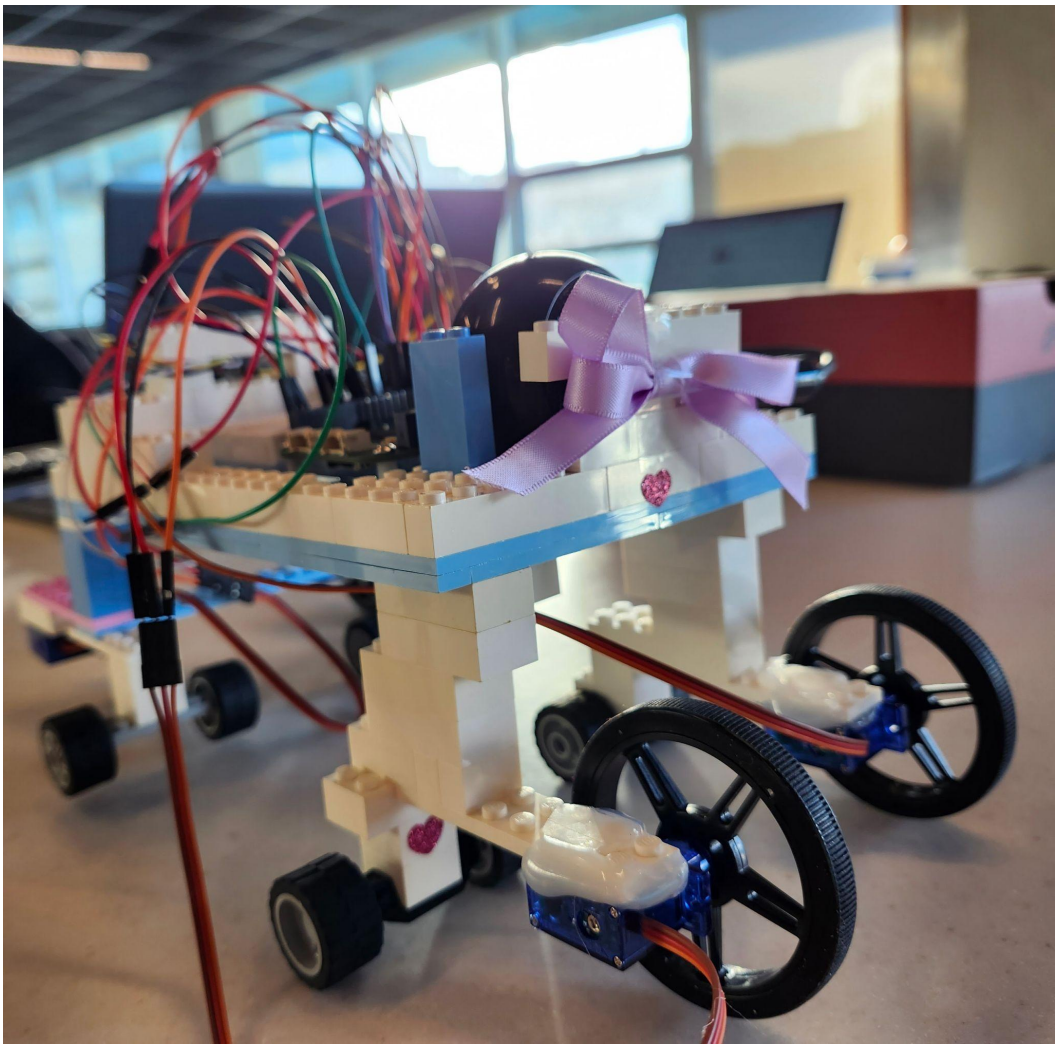
December 7, 2023

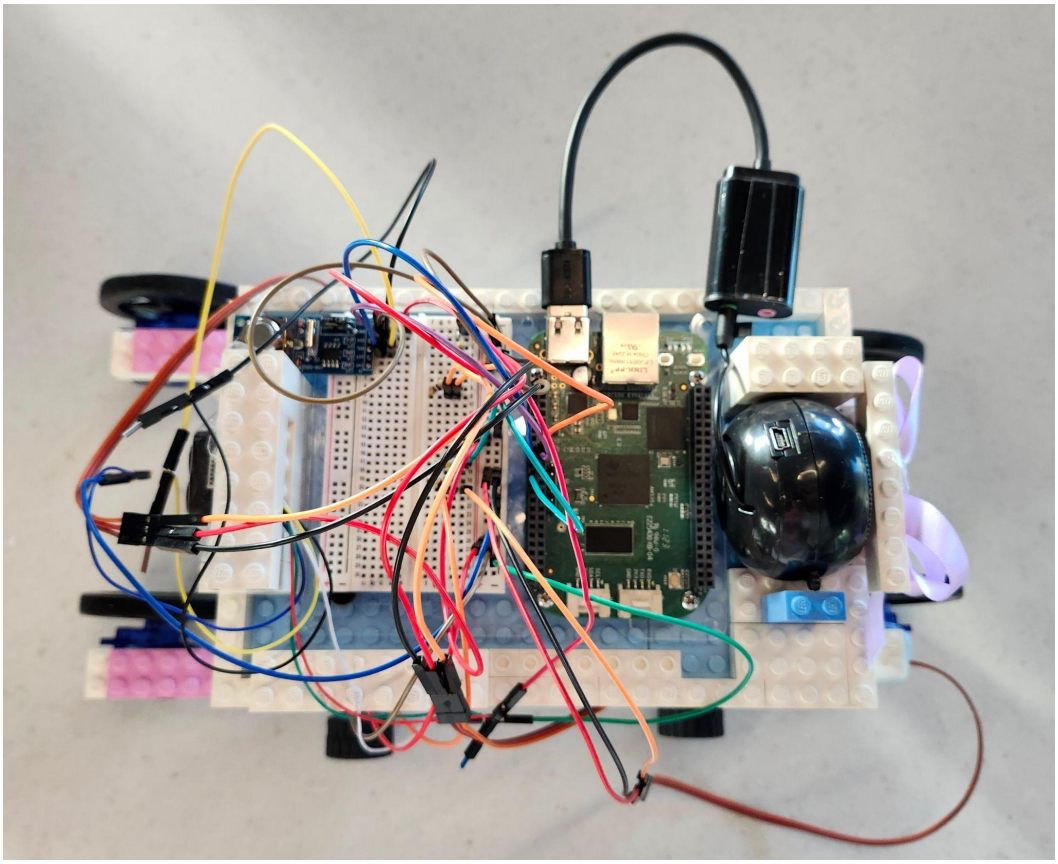
Project Write-Up

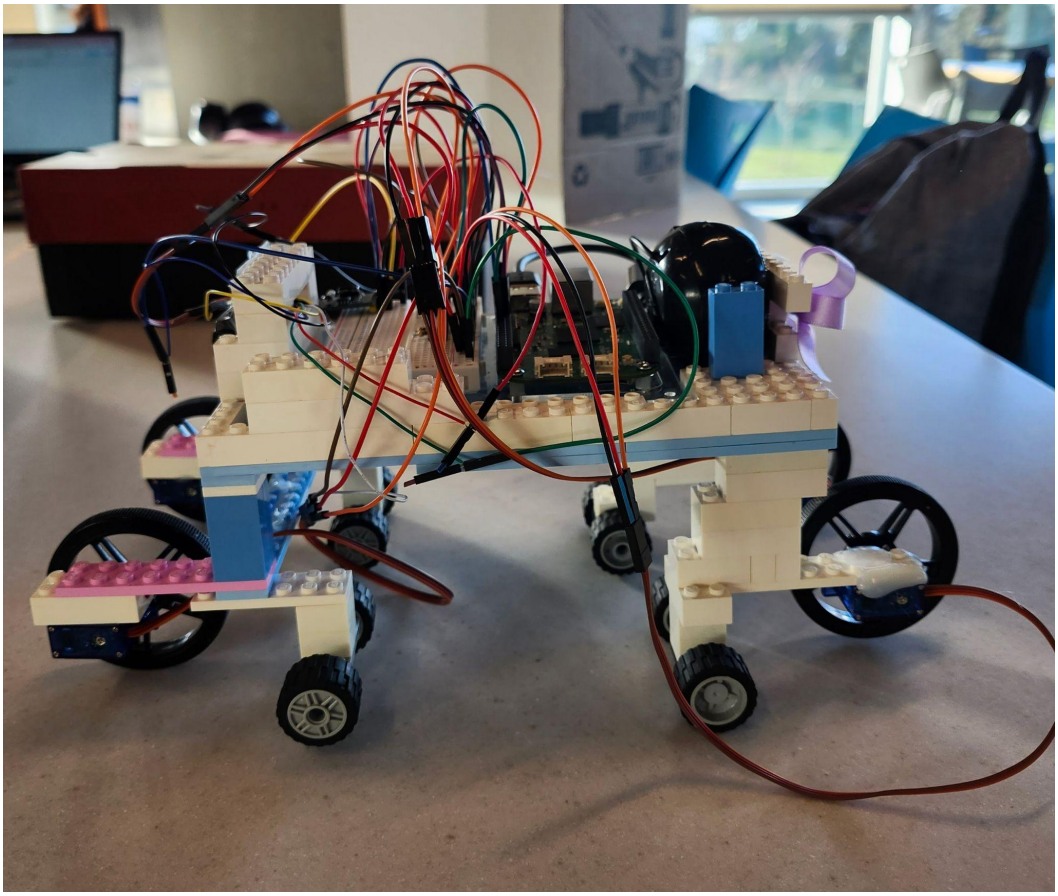
System Explanation

As university students who are prone to snoozing alarms and sleeping in, our group decided to build a moving alarm clock that will continue to sound and move around the room until the user gets up and presses a button to turn it off.

Our system is an alarm clock with wheels that will move around once the alarm goes off, and keep moving until the user presses a button to turn it off. Our system has a sensor at the front to sense objects and obstructions in its path, multiple servo motors attached to wheels to make it move, a real time clock to check the time and know when the alarm should go off, and a speaker to play the alarm sound.







Once it reaches the time for the alarm to go off, the speaker will start playing our selected audio as the alarm. At the same time, the wheels will start moving and our project will begin moving around. There is a TF-Luna LiDAR sensor at the front of the alarm clock that continuously reads how far away objects are from the sensor. This sensor guides our alarm clock, so when the clock reaches a certain distance from an object - 20 cm- it will back up to avoid the obstruction. The alarm will keep sounding and the clock will keep moving until it reads input from the user, specifically the user must press the user button for the alarm to turn off and stop moving.

Important things not working well

Pulse-Width Modulation (PWM):
The original way to get the motors spinning to make our project move involved using PWMs. Servo motors move based off input frequency, which is why we need the PWM to send the signals. However, the PWM was not connecting on our boards, and we talked to the TA's as well as the prof to get their input but no one had any solutions. We ended up using the GPIO pins to connect our servo to our boards and get our project moving. As the GPIO can only do 1 or 0 for input/output, we wrote code that would output 1, sleep, then output 0 to imitate a duty cycle to get our motors moving.

Real Time Clock (RTC):
A real time clock is used to measure and keep time. This device has two modes, countdown mode and alarm mode. Ideally, for our project we would use alarm mode to set the alarm for a specific time and day. However, we could not get that working. Instead we used countdown mode, so the alarm starts going off at the end of the countdown. Unfortunately, in countdown mode we could not get the timer flag working. Once the countdown ends, a bit should turn on and this flags that the countdown is over. Instead we manually check the countdown to see if it's ended, and once this is true, the alarm begins to play.

Servo Motors:
Our robot can move, but only forwards and backwards. We tried using 180 degree servo motors to create an axle to be able to turn our robot but it did not turn properly. We broke both of our 180 degree motors trying to fix the problem, so in the end we decided to only use our continuous 360 degree motors and leave it as a robot that can only move in the y-axis.

Feature Table

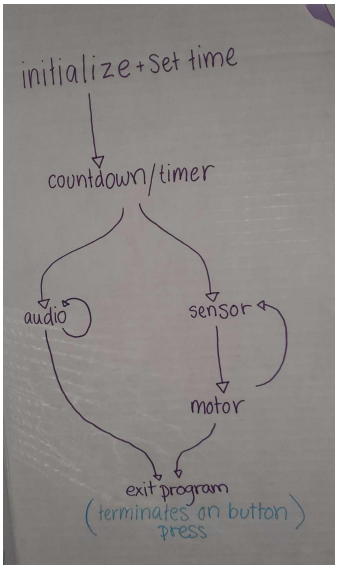
Description	Output	Completeness	Coding lang.	Author(s)	Notes
TF Luna Sensor	T	1	C	Eugenia	Reads path and recognizes obstructions
Servo Motors	T	2	C	Qiraa	Moves backwards and forwards, cannot turn
Speaker	T	1	C	Eugenia	Plays the alarm sound at the correct time
Real Time Clock	T	2	C	Sundus	Using countdown mode, instead of alarm mode to set specific time

Extra Hardware and Software Used

In addition to our Beaglebone Green board that hosts our project, we used a number of other hardware and software libraries.

In terms of hardware, we used a real time clock, a speaker, a sensor, and the servo motors. There is also a USB audio adapter we needed to connect the speaker to our beaglebone board, as well as a battery pack to power our project.

For our servo motors, we asked the TA's for help with writing code to allow the wheels to spin and make our project move.



This is a fork diagram to give an overview of how our code for the robot functions, and in what order everything happens in.