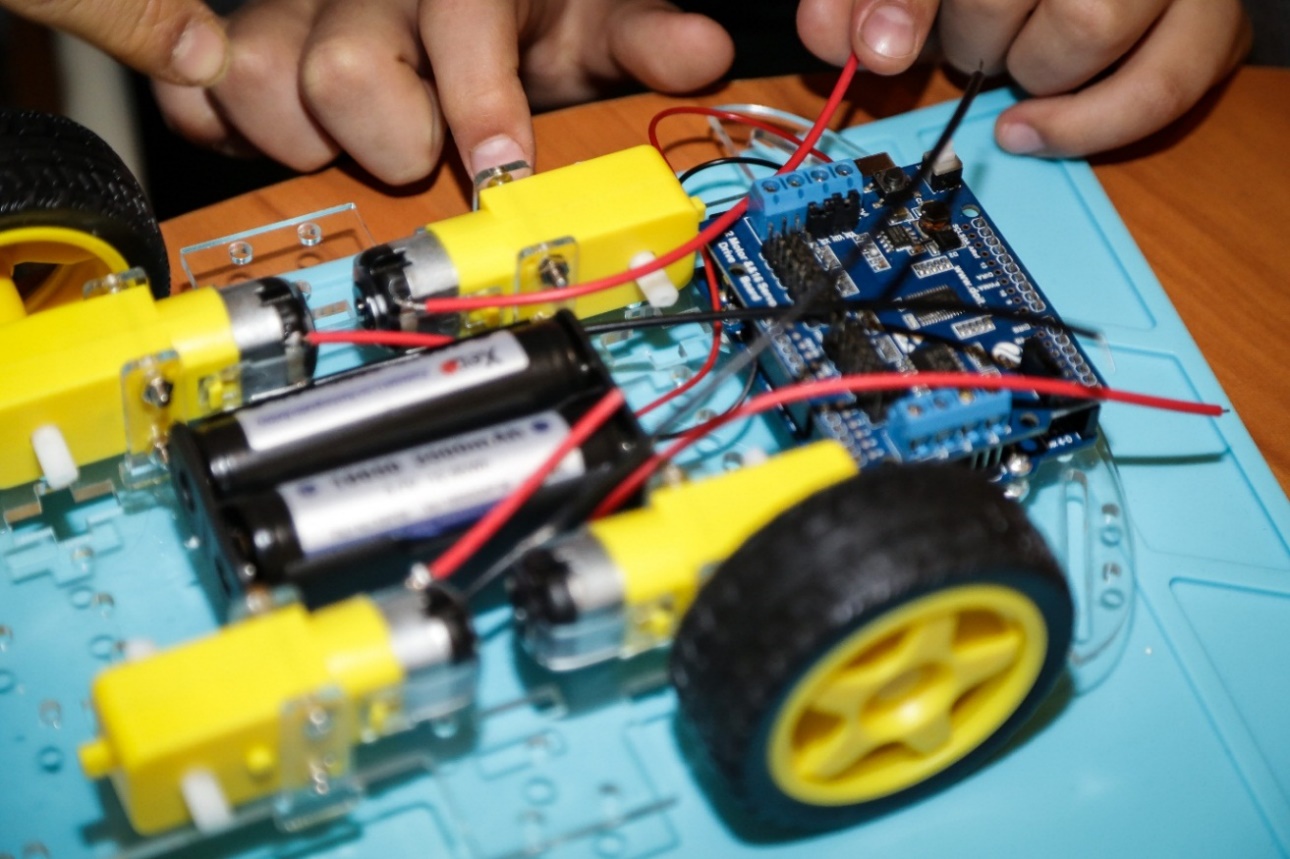
**Technical document**

Robot makers

**Technical Document for the team Robot Makers**

Brief Overview:

We will have 3 robots which will demonstrate the things we are saying, with snippets of code projected onto the whiteboard. One of these robots will have its costume taken of so that all of the wiring and main structure is exposed for people to see and so that we can refer to/show certain features of the robot.

*Hardware*

* Sensors
  + **Ultrasonic-** By making the robot wait until it’s activated with the ultrasonic sensor, we increase safety, since as soon as we plug in a robot to program it, it will start which means the robot could drive into something or burn the computer if it can’t handle the current the robot is drawing. Furthermore, while the robot is moving, the controller is constantly checking the ultrasonic sensor to see if there is anything in front of the robot (such as a human), if there is, it will wait until the object is going before continuing forward (or turning).
  + **Gyro-** Previously for turning, the method we used was to find the time it takes to turn 180 degrees, then divide by 180 and multiply by the angle we want the robot to turn. The problem was that all of the motors, although technically the same could still turn at different speeds (or if the batteries are low, meaning the motors will spin slower) The impact of which is that the robot would fall out of sync as well as sometimes not turn enough or overturn. Overall, this was very unreliable so we decided to add a gyro sensor which makes sure that no matter how fast or slow or inhibited the motors are, they robots will always turn more or less to the exact angle needed consistently (since the robot will turn until the input angle is equal to or greater than the gyro angle).
  + **Colour-** The colour sensor was added so that the robot would not fall off the stage during the performance. We found the values for black and white colours, then we added some margin for error and made the controller check that the colour sensor is reading white, if it’s not then the robot will move back until it’s a safe distance away from the edge. This is another safety feature which will save hours of work and lots of money.
  + **Radio and then Bluetooth module-** We used to use a radio module for robot to robot communication. We would set them to send and receive on the same frequency and at the start of the performance one of the robots waits to receive the string “start” which the other robot will send once it’s activated using the ultrasonic sensor. However, since the rules at this competition don’t allow us to use radio, we decided to use Bluetooth to do the same functions. Furthermore, we used the robot to robot communication to sync up our robots throughout the dance, since one of the robots was usually a bit faster. This piece if hardware made sure that the robots were more reliable since they were doing the same moves at the same time, which reduces the chances of an accident. This piece of technology also meant that we don’t have to try to activate both robots at the exact same time, which would be inconsistent.
* Quality of life/Reliability
  + **Switch-** On one of the previous competitions, the switch on the motor controller broke and meant that the robot would turn off randomly, because of this we had to solder the pins of the switch together so that we had to take the batteries out and put them back in, to turn the robot on and off. To ensure we didn’t have this problem again, we added our own switch made of metal and one that cannot easily break. We also placed it in an easy to access position which is a nice bonus.
  + **Ready LED-** The gyro sensor needs time to calibrate, during this time the robot shouldn’t be moved and therefore we added an LED as a visual prompt that shows us that the robot is ready to dance. This was also useful for troubleshooting.
* Signal processing of sensor data
  + **Pulse Width Modulation (PWM)** works by varying the amount of time that an electric pulse is on/off for. By doing this it controls the amount of current that goes to an output device. We use this to control the speed of our motors, the longer the amount of time that the pulse is on, the faster the motor will spin. Furthermore, the time between pulses is so short that the motor will constantly spin.
  + **Analogue data** is used for sensors which have a constant fluctuating output that can be a large range of numbers. We use this for our colour sensor which can output many different values for different colours, the ultrasonic sensor which outputs a varying value for distance, the gyro sensor which has a variety of different angles.
  + **Digital data** is used for sensors which have 2 outputs that can be one of two values. We use this for our LED which only has two states (on [high] or off [low]).

*Software*

* **Software architecture-** We started off by making functions for moving and turning to highly simplify code and make it much more efficient to program the robots. Using this method, instead of writing five lines of code every time we wanted to move a robot we could simply write 1 line of code. Furthermore, we moved these functions into a library which allowed us to simplify our code further by not having to declare pages of functions at the start of every program, we could simply import the library with one line of code and start using the function.
* Our robots use multiple **feedback loops**. While driving forward or turning, the ultrasonic will be measuring distances and if it detects any object within a certain distance it will wait before moving forward so that the robot will not collide with it. We also use a feedback loop when turning using the gyro sensor. The controller will check what angle the gyro is at until it reaches the set angle. A HUGE problem that we ran into while trying to implement these feedback loops was that we used the delay function to make the controller wait for the previous action to finish the next one. Consequently, we had to create our own function that would measure the time since the previous action while checking the output from the sensors.

*Communication*

* We have **human to robot** communication with the ultrasonic sensor which will activate the program whenever we want to.
* We have **robot to human** communication with the LED that indicates when the robot is ready to dance.
* We have **robot to robot** communication with the Bluetooth module that activates the second robot as soon as the first robot is activated. Not only that but it also allows the robots to sync up throughout the dance. The ultrasonic is also used for robot to robot communication as it will check if they are going to crash and stop them from doing so.

*Subsystems*

* We have the **ultrasonic sensor** constantly checking if there is an object in the way of the robot while it’s moving forward or turning. If there is, the robot will wait or the object to move out of the way before containing to dance. This increases safety as there will not be a collision and therefore neither the robot nor object (which could be a person) will be harmed or damaged.
* The **colour sensor** is constantly checking if the robot is at the edge of the dancing area throughout the dance. If it detects the black line which marks the boundary of the area, it will move back until it’s a safe distance from the edge. This means that the robot will not fall off the side of the stage and prevent the damage that this would inflict.
* We have a **switch with an LED** which is a system for making sure the robots are ready for their performance. When we turn on the switch we wait until we see the visual cue that the robots are ready to dance. This way the robots will not let us down since we know that they will start.

*Challenges*

* **Reliability-** this was a big problem as we couldn’t reliably do our performance multiple times in a row. There were many factors which contributed to this and the following are the fixes we implemented, and which have made our robots so much more reliable.
  + We added a **gyro sensor** to the robot. The unreliable part was when we used to use the time it takes to turn one degree multiplied by the angle we wanted it to turn. This didn’t work well and was constantly under or overturning which sometimes ruined the dance routine. However, once we added a gyro sensor, the amount of error was reduced significantly (to a few degrees) and it also wasn’t affected by different motor speeds as both of the robots would always turn to the exact same spot.
  + Throughout the dance we would make the **faster robot** stop and wait for a command from the slower robot via Bluetooth before it started moving so that they would be moving together. Before, one of the robots would get ahead of the other one and when they got to a part where they needed to do some moves in sync, they wouldn’t be and this made the performance look much worse.
  + At the national competition the **switch** on motor controller would randomly fall out and turn the robot off midway through the performance. This was a frustrating problem and we had to solder the switch’s pins together and take the batteries out or in to turn the robot on or off. To make sure this didn’t happen again we added our own (metal instead of plastic) switch which is much harder to break and is much more reliable.
  + Our minions’ **arms** would sometimes **fall off**, therefore we had to screw the arms into the motor which fixed it.
* **Finding the right balance**
  + Making **everything work together** (Software, hardware and electrical)- Finding the right balance was quite difficult and took us over 7 versions of our minions to get right. The first version was very unstable and would just fall over whenever it moved, so we decided so make the base bigger (Which also made it heavier) and although this one didn’t fall over the motors could barely handle the weight which meant that we needed bigger motors, which need more batteries, which also means a stronger motor controller, etc.

With the **electrical side**, wiring was quite difficult since we had to find optimal paths for the wires to get from all around the minion, into the motor controller, without interfering with the hardware. For example, when we were able to connect wires from the motor to the motor controller, however the motors still didn’t work, this was because the wires were too short and were being stretched.

Between **software and hardware**, we had to **compromise**, since we made a dance routine for the robot while it only had the base. Then once we put the whole robot together it turned out that the routine was WAY to fast and had too many complicated moves that didn’t suit out robot. So we simplified the code. Furthermore, we had to make the software and hardware work together since previously we couldn’t check sensor readings while the robot was moving. Therefore, we created our own function.

*Conclusion*

* Overall, we have a **well-rounded** robot which **reacts to its environment**, communicates with other robots, can move quite fast and is agile, however is a little wobbly on turns, however is very precise since the gyro sensor is used. Basically, a robot waits until a human activates it, then sends a command via bluetooth to the other robot to start and they dance, all while checking if there is anything in front of them, if there is, they will stop. Throughout the dance they sync up since the fast one waits till the slow one catches up and sends a command.