However, since the rules at this competition don’t allow us to use radio, we decided to use Bluetooth to do the same functions but were having trouble pairing the Bluetooth modules with the Arduino board. We use PWM (Pulse Width Modulation) signals, which work by varying the amount of time that an electric pulse is on/off for in a series of pulses. By doing this it controls the amount of power 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. **We chose** to use the Arduino micro-controller since it is very capable and can perform a lot of functions as well as the fact that it is highly customisable with many resources available for use.

Hello, Gru told me that you wanted to know more about the minions from the team ‘Robot Makers’. So, let’s start with the hardware. Our robot has an ultrasonic sensor that we use to make the robot wait at the start of the program until it’s activated by putting something in front of 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 needed. 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 or a minion), if there is, it will wait until the object is gone before continuing on.

consistently turn very close the exact angle needed (since the robot will turn until the input angle is equal to or greater than the gyro angle) **\*\*\***. **Another** sensor that we have is the colour sensor. We added this sensor so that the robot would not fall off the stage during the performance. We found the values for black and white colours (which we can adjust in different environments thanks to our calibration function), 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. **\*\*\***

Next we have the gyro sensor. Previously, the method we used for turning was to find the time it takes to turn 180 degrees, then divide that 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 robots 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 the motors are, the robots will

**The robots** have all forms of communication, human to robot (with the ultrasonic sensor which will activate the program whenever we want to), robot to human (with the LED) and robot to robot (with the ultrasonic that checks to make sure they won’t crash into each other). **Our robots** also have a few subsystems including the **ultrasonic sensor** constantly checking if there is an object in the way of the robot while it’s moving forward or turning . The **colour sensor** is constantly checking if the robot is at the edge of the dancing area throughout the dance and finally the gyro sensor which checks that we are turning to the correct angle. On the **software side** we started off by making functions for moving and turning to highly simplify code and make it much more efficient to program the robots.

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 then 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. We also added an LED to indicate when our robot is ready to perform **\*\*\***. Since the gyro sensor needs time to calibrate and during this time the robot shouldn’t be moved. It was also useful for troubleshooting.

We had a radio transmitter and receiver for robot to robot communication. We would set one robot to send and the other to 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, meaning they will start at the same time. 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 of 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 as well as making our performances more reliable.

batteries, which also means a stronger motor controller, etc. On 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 our 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. Which led us to create our own function. 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, but is very precise since the gyro sensor is used.

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. However, we didn’t want it to wait, we wanted it to be checking the sensors. Consequently, we had to create our own function that would measure the time since the

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 since we could simply import the library with one line of code and start using all of the functions. On top of this a library allows us to make a change in its program which will carry through to all of the other programs that use the library (instead of having to change it in all of them) as well as the ability to use object oriented programming to use different values for parts of different robots that we get from calibration. **Our robots** use multiple feedback loops.

The **switch** on the motor controller would randomly fall out and so we added our own switch after soldering the pins of the old one together. Our minions’ **arms** would sometimes **fall off**, therefore we had to screw the arms into the motor which fixed it. **Finding the right** balance and making everything work together (Software, hardware and electrical) was also a big challenge and it 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

previous action while checking the output from the sensors. **Reliability** 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 consistent. We added a **gyro sensor** to the robot. The unreliable part was when we used time (as mentioned before). 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.