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),

which reduces the chances of an accident as well as making our performances more reliable. 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 we’re having trouble pairing the Bluetooth modules with the Arduino board and so we decided to use an infrared transmitter and receiver to make the robots start in sync (one will wait until the other is activated using the ultrasonic sensor). **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, since the longer the amount of time that the pulse is on, the faster the motor will spin

if there is, it will wait until the object is gone before continuing on.

**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 (especially if the batteries are low) The impact of which is that the robots would fall out of sync as well as sometimes not turn enough or overturn. Therefore, we use the gyro to make sure that no matter how fast or slow the motors are, the robots will consistently turn very close the exact angle needed **\*\*\***.

**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. **\*\*\***

. **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 meaning that we can build our robot from scratch to perform any function that we want, instead of using a preset kit with limited capabilities. **The robots** have all forms of communication, human to robot (with the ultrasonic sensor), 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). 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 plastic 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 to turn the robot off. To ensure we didn’t have this problem again, we added our own metal switch. 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. One robot would send and the other received 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 made sure that the robots were doing the same moves at the same time

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 stronger batteries which also means a stronger motor controller, etc. 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 with the gyro sensor.

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. Then, 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 that will carry through to all of the other programs that use the library (instead of having to make the change in all of them) as well as the ability to use object oriented programming to use different values of sensors and motors on different robots, using the same library.

**Our robots** also use multiple feedback loops. While moving, the ultrasonic will be used to detect objects within a certain distance which will make it wait before moving to avoid collision **\*\*\***. 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 before going on to 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 do this. **Reliability** was a big problem as we couldn’t consistently do our performance. There were many factors which contributed to this and the following are the fixes we implemented. We added a **gyro sensor** to the robot so we didn’t have to use time. The **switch** on the motor controller failed, so we added our own switch. **Finding the right** balance and making everything work together was also a big challenge and it took us over 7 versions of our minions to get right.