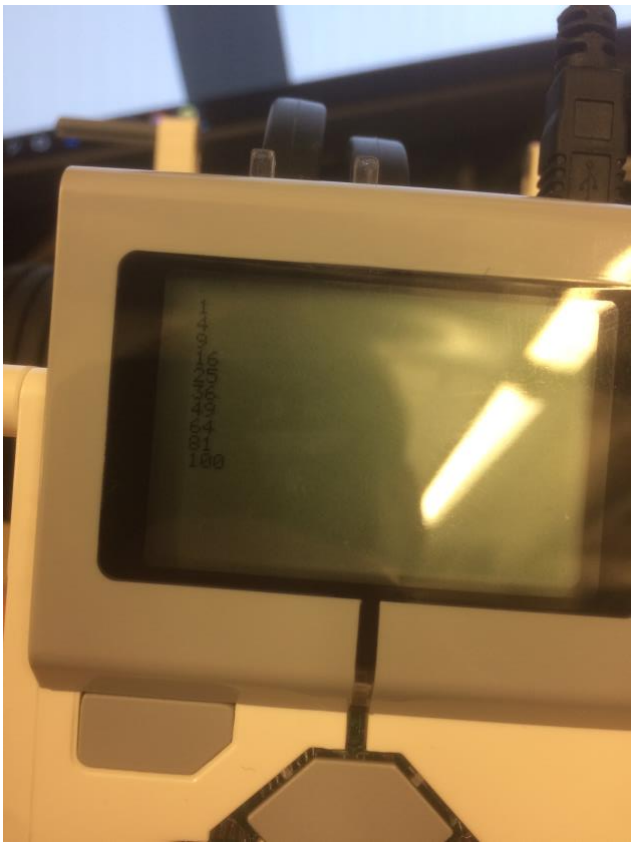
Matthew Donnelly
Daniel Bjorklund

Lab #1 report: team 2

1.

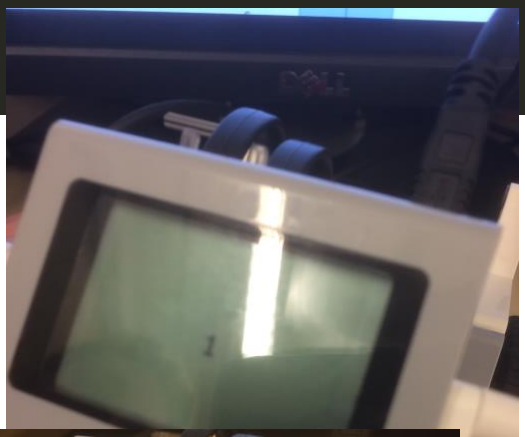
```
#pragma config(StandardModel, "EV3_REMBOT")
task main()
{
    int i;
    i=1;
    while (i<=10){

        displayTextLine(i,"%d",i*i);
        i++;
    }


    sleep(10000);


}
```

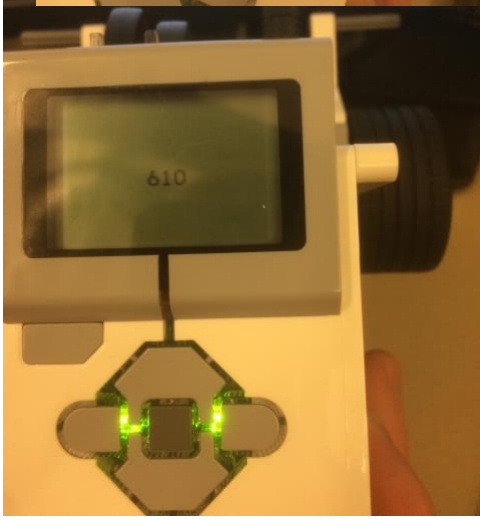


A simple program, it loops ten times, each time calculating the square of that i and displaying it on the i-th line. After it displays each, it sleeps for 10 seconds before the program ends.

2.

```
#pragma config(StandardModel, "EV3_REMBOT")
task main()
{
    int first;
    int second;
    int answer;
    int i;

    first = 0;
    second = 1;
    i=0;

    while (i<20){

        answer = first + second;

        if(i<2){
            displayCenteredBigTextLine(8, "%d", i);
        }
        else{
            displayCenteredBigTextLine(8, "%d", answer);
            first = second;
            second = answer;
        }


        sleep(1000);
        i++;
    }


}
```





Comments for Q2: Although recursion is quicker to program for the Fibonacci sequence it usually calculates more than necessary. Because the lego robots have limited resources we wanted to do it more efficiently by using a normal while loop. If i is 0 or 1 it displays i, otherwise it calculates it, storing the previous two results in the corresponding variables. It also sleeps for 1 second every iteration so that the results is visible on the screen.

3.

```
#pragma config(StandardModel, "EV3_REMBOT")

void turn(){
    setMotorSync(leftMotor, rightMotor, 99, 35);
    sleep(1000);
}

void go(){
    setMotorSync(leftMotor, rightMotor, 0, 50);
    sleep(2250);
}

task main()
{

    go();
    turn();
    go();


}
```

Comments: All the program does is go the required distance, turn around and go back. Figuring out the values for the turn and go functions is described below.

Challenges faced/unanticipated behavior:

Our main challenge was figuring out the proper values for turning, primarily the motor power percentage (35% in this case). We discovered that depending on what surface we tested on it would turn for different amounts due to the varying friction of the surfaces. On the carpet the tires had less grip and it wouldn't turn completely; on the desk the tires had too much grip and would over turn. To combat this, we only tested on the surface that the final run would be done on.

Figuring out the sleep time for how long to move forward was much less challenging. We did it by trial and error. Started at 2 seconds and it did not go far enough. Moved it to 2.5 seconds and it went too far. Then we split the difference and made it 2.25 seconds and it was just right for the distance.

4.

```
#pragma config(StandardModel, "EV3_REMBOT")

void turnRight(){
    setMotorSync(leftMotor, rightMotor, 99, 38);
    sleep(500);
}

void turnLeft(){
    setMotorSync(leftMotor, rightMotor, -99, 38);
    sleep(500);
}

void go(int x){
    setMotorSync(leftMotor, rightMotor, 0, 50);
    sleep(x);
}

task main()
{
    go(1912);
    turnLeft();
    go(2040);
    turnLeft();
    go(1173);
    turnLeft();
    go(306);
    turnRight();
    go(2040);
    turnLeft();
    go(1912);


}
```

Comments: Similar to the previous question, this program is specifically for going through the designed course. For both turn functions we kept the same values as the turn from question 3 but halved the sleep time so it would turn 90 degrees instead of 180. We also changed the sleep in the go function to be a parameter.

Challenges: Our main challenge was figuring out how long to sleep during the ‘go’ function. We knew from the previous question that it would travel 23 inches at 50% power over 2.25 seconds. Using this info, we derived that it would travel an inch in ~97 milliseconds. We then just multiplied the number of inches required for each leg by 97 and got the sleep values used.

The power level we used in this program (as well as the others) was a random choice that we went with. Once we figured it out for that power level it was easier for us to stick with it. The way we derived the sleep values was fairly systematic for our current power level, but translating this across power levels is another process entirely. But one power level I think figuring out how long it takes to go a specified unit (an inch) and then multiplying it out is a quick and easy way to get fairly accurate values.

Extra programs you asked Daniel and I to make after we finished

Make a program that counts up and down when you hit left and right

```
task main()
{
    int i;
    i =0;
    displayCenteredTextLine(1, "Current Value:");

    while(true){
        if (getButtonPress(buttonRight)){
            i++;
            displayCenteredBigTextLine(4, "%d", i);
        }
        else if (getButtonPress(buttonLeft)){
            i--;
            displayCenteredBigTextLine(4, "%d", i);
        }

        else if (getButtonPress(buttonUp)||getButtonPress(buttonDown)){
            return;
        }









        sleep(350);
    }


}
```

Make the same program using different tasks for display and the math

```
int i;

task math()
{
    i=0;
    while(true){
        if (getButtonPress(buttonRight)){
            i++;
        }
        else if (getButtonPress(buttonLeft)){
            i--;
        }
        sleep(250);
    }

}

task display()
{
    displayCenteredTextLine(1, "Current Value:");

    while(true){

        if (getButtonPress(buttonUp)||getButtonPress(buttonDown)){
            stopAllTasks();
        }

        else{
            displayCenteredBigTextLine(4, "%d", i);
        }
        sleep(250);
    }

}

task main(){
    startTask(math, 10);
    startTask(display, 10);
    while(true){}
}
```