Project 3
Team 43

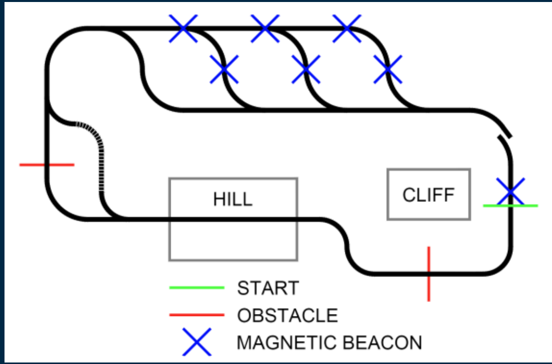
Lily Crouse
Gabe Kurfman
Braden Seasor

Project Objectives

We have been tasked with the development of a Mars Cargo Rover (MACRO) prototype out of a provided kit of Legos materials and a variety of Pi bricks. MACRO must be capable of the following:

- "Precise navigation to specific sites”
- “Recognition and handling of hazards”
- “Timely delivery of mission hardware”
- “Transporting cargo from location to location without dropping or tipping"

-Project 3 Description Fall 2022



Project Management

Metrics of Success

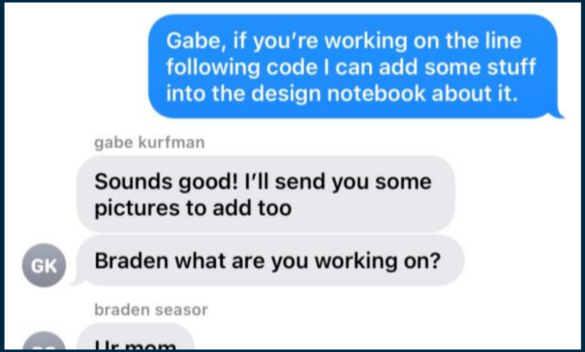
- PoC task execution
- Checklist of listed tasks in project description
- Consistency rates

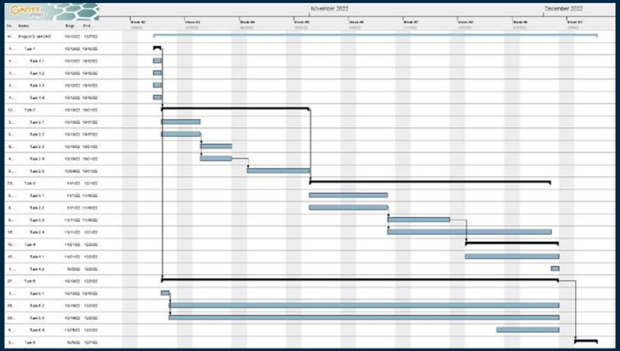
Task Delegation

- Gabe developed code
- Decisions were made as a group
- Execution was decided on task-by-task basis

Time Management

- Met at least once a week, tried to make meaningful decisions within each meeting
- Gantt Chart





Our Process

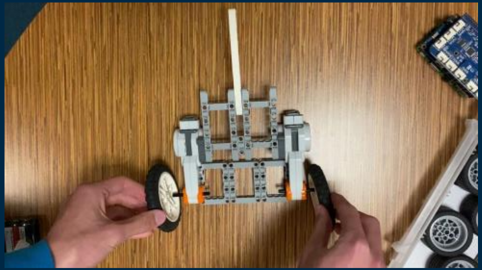
MID OCTOBER
INITIAL PLANNING
Prototyping with LEGO parts

EARLY NOVEMBER
FIRST TESTING
Attempting line following

MID NOVEMBER
REVISIONS
Optimizing code and handling obstacles

DECEMBER 1st – 3rd
DRASTIC CHANGES
Major revisions to handle massive obstacles

Design Decisions – Steering System

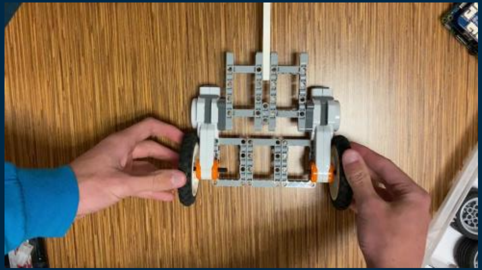


Pointed Steering

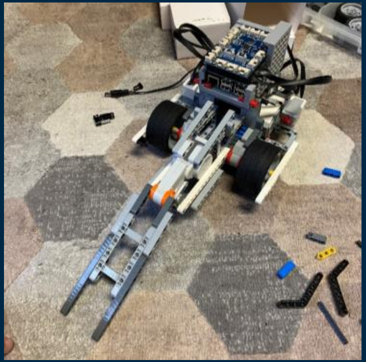
- More efficient at high speeds
- Tried and tested in modern cars

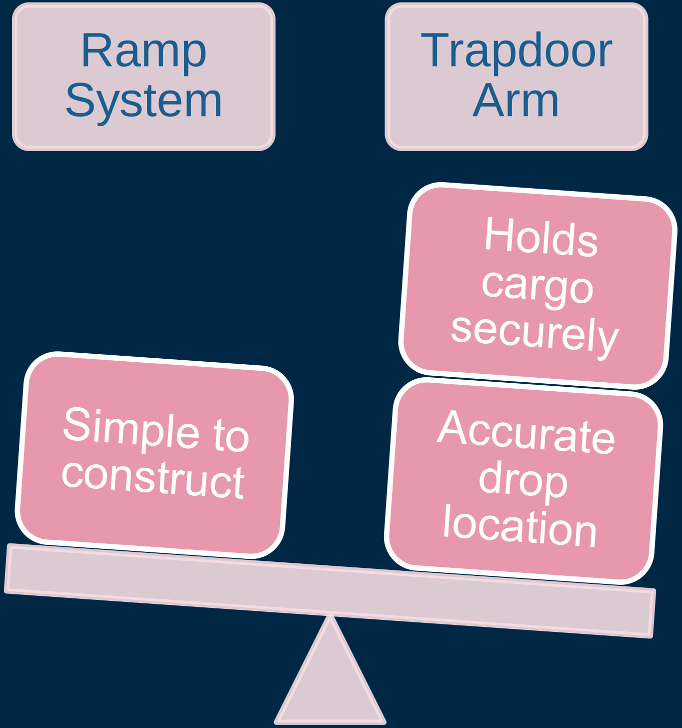
Tank Steering

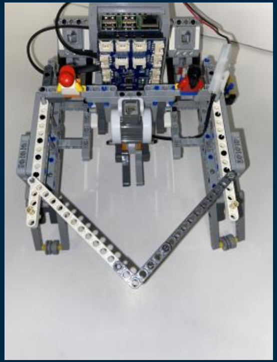
- Less moving parts
- Very small turn radius
- Uses one less motor



Design Decisions – Cargo Delivery







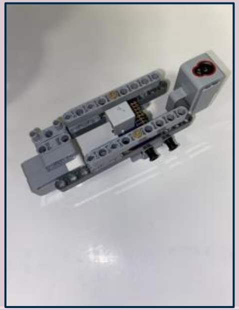
Alternate Dropping Mechanisms

- Conveyor System
- Ramp and Gate
- Elastic Arm System
- Trapdoor System

Other Design Features

Sensor Array

- Compact
- Easy to modify
- Gyro, Color, & Hall Sensor

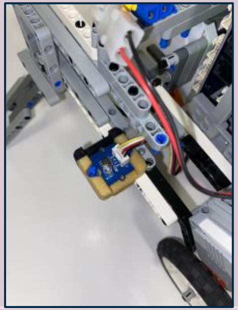


Battery Compartment

- Easy to remove for charging
- Secures battery

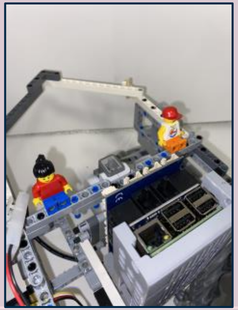
Light Sensor

- Allows for easy user input
- Hands free

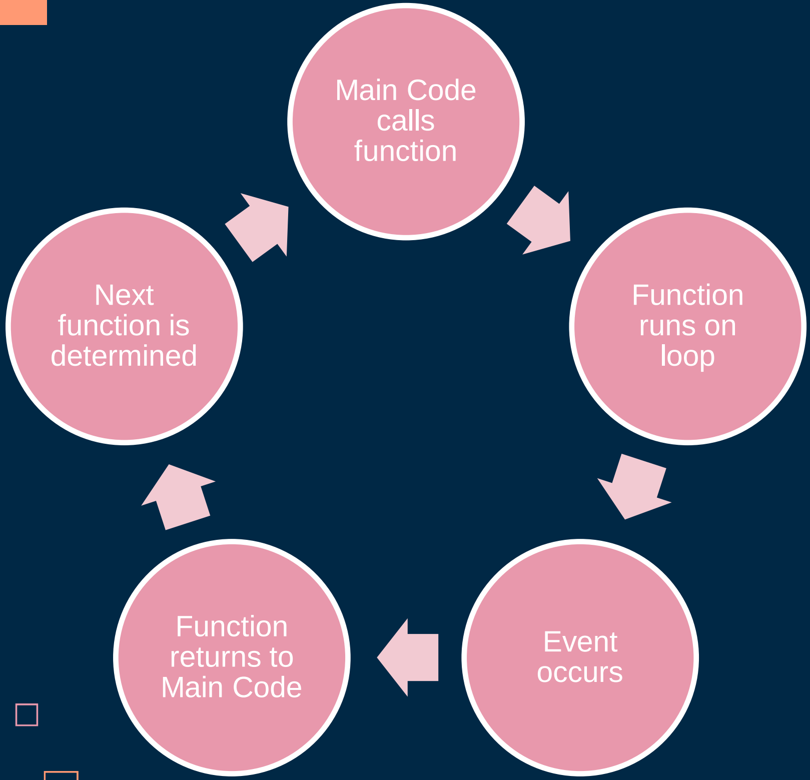


Pilots

- Cute
- Necessary component
- Guide robot to success



The Build Process – Software and Logic



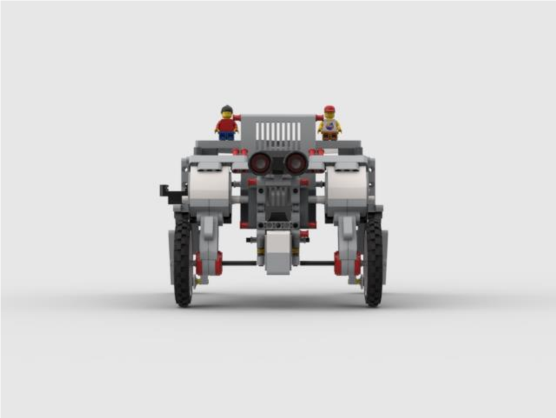

```python
# Follow line to path split
for i in range(0, path):
    followLineMag(side="left", speed=0.5, accel = a)
    time.sleep(0.25)
    followLine(0.5, "left")
```

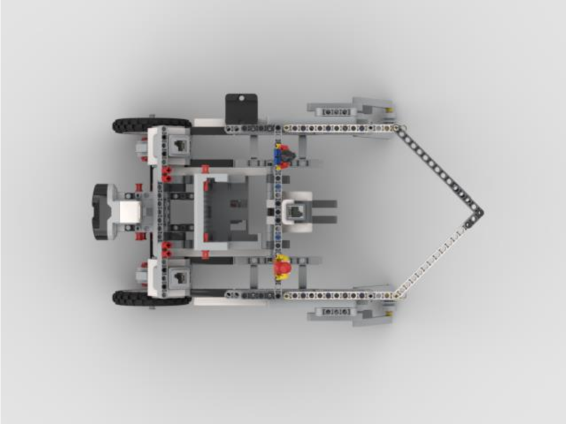
```python
def followLineMag(side = "right", accel = 1.001, direction = 1, speed = 0.5):
    """

    Args:
        side (string, optional): "right" for right-side line following.
        accel (float, optional): Rate of turn speed increase if missing line.
        direction (int, optional): 1 for forward, -1 for reverse. Defaults to 1.
        speed (float, optional): Speed to drive in rotations/s. Defaults to 0.5.
    """
    print("%s-side line following at %.2f rotations/s until magnet detected."
            % (side.capitalize(), direction * speed))

    sweepAngle = 90
    if (side == "right"):
        sideInt = -1
    else:
        sideInt = 1

    BP.set_motor_dps(RIGHT_MOTOR+LEFT_MOTOR, speed)

    defaultTurn = 0.6
    turnRate = defaultTurn
    sweep = False
    angle = 0
    angleDirection = 1
    offsetAngle = BP.get_sensor(GYRO_SENSOR)[0]
    correctedDirection = 1

    severity = 0
```

```python
    while (not magnetDetected()):
        color = BP.get_sensor(COLOR_SENSOR)

        severity = -turnRate + (2*turnRate) / (WHITE - BLACK) * (color - BLACK)
        severity *= correctedDirection

        if (angleDirection == sign(severity)):
            try:
                angle = BP.get_sensor(GYRO_SENSOR)[0] - offsetAngle
            except brickpi3.SensorError:
                pass

            if (abs(angle) > abs(sweepAngle)):
                sweep = True
                correctedDirection = -sign(angle)
                angleDirection = correctedDirection
                turnRate = defaultTurn / 2

            # Slowly ramp up turn
            elif (turnRate < 1 and not sweep):
                turnRate *= accel

        else:
            if turnRate > defaultTurn:
                turnRate = defaultTurn

            angleDirection = sign(severity)
            offsetAngle = BP.get_sensor(GYRO_SENSOR)[0]
            correctedDirection = 1
            sweep = False
```

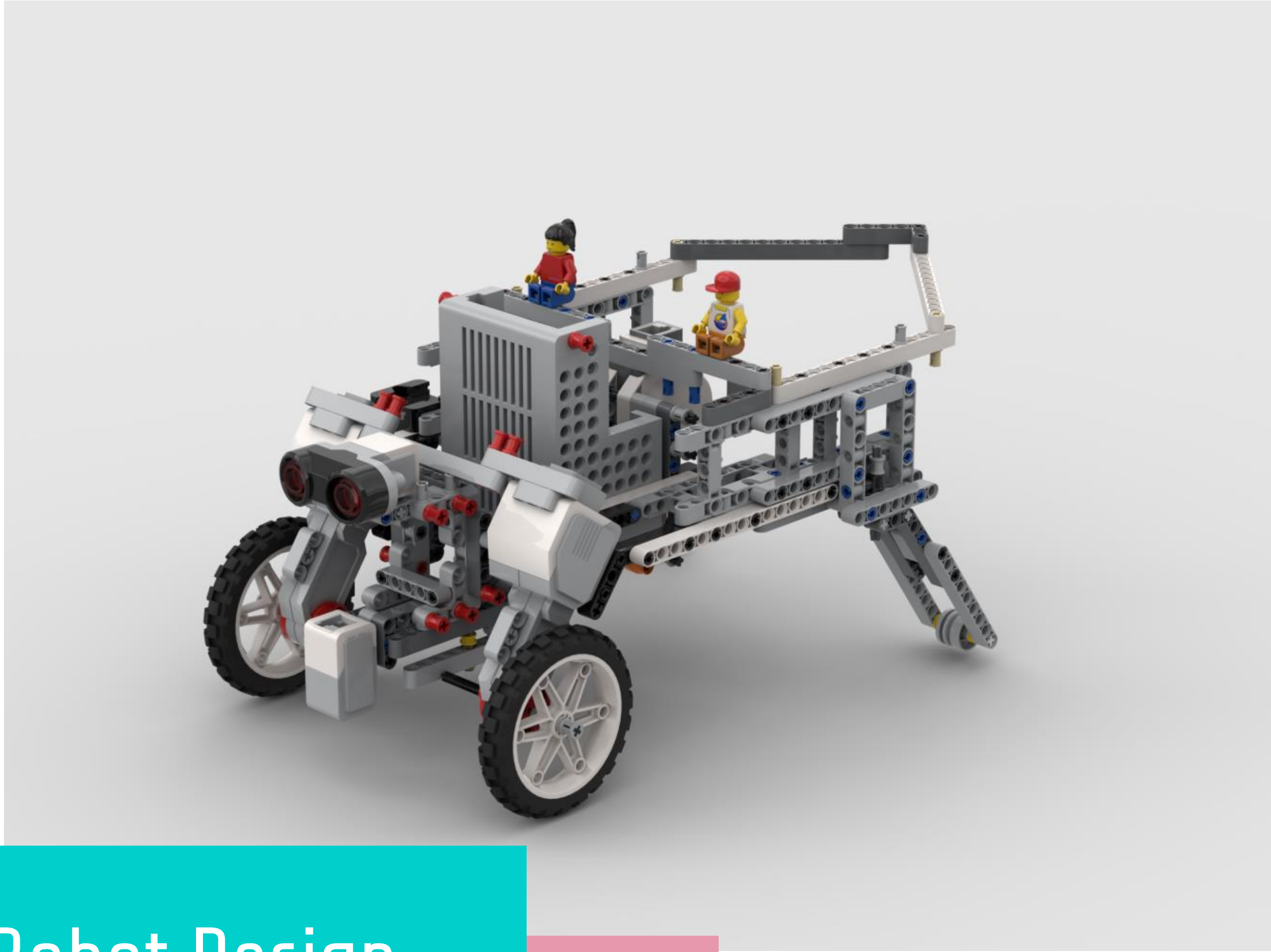
The Build Process – Software and Logic

Sensor Calibration Functions	Sensor Reading Functions	Drive Motor Functions	Cargo Motor Functions	Main Code
• calibrateColor • calibrateMagnet • calibrateLight	• magnetDetected • lightDetected • obstacleDetected • waitForLight • printDebug	• resetMotors • driveMotors • turnMotors • angleMotors • followLine • FollowLineMag	• cargoMotor • lockCargo • unlockCargo	• Follow line to correct magnet • Turn down desired path • Drop cargo on target • Return to start





Final Robot Design



Design Specs

Unloaded Weight: 630 g

Size: 23 x 38 x 21 cm

Total Parts: 293

Most Common Part:

Black Technic Pin (x83)

Final Robot Design

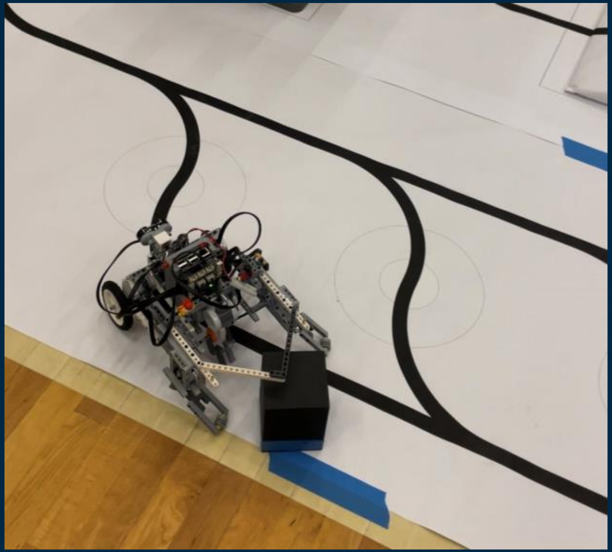
Positives

- Superior ability to scale terrain
- Superior ability to carry cargo
- Increased stability
- Increased consistency with speed
- Refined depositing mechanism

Time (s)	Distance (cm)	Speed (cm/s)	Desired Speed (cm/s)	% Error
1.95	30	15.38	15	2.56
2.02	30	14.85	15	-0.99
1.8	30	16.67	15	11.11
1.95	30	15.38	15	2.56
1.57	30	19.11	20	-4.46
1.47	30	20.41	20	2.04
1.57	30	19.11	20	-4.46
1.43	30	20.98	20	4.90
1.18	30	25.42	25	1.69
1.23	30	24.39	25	-2.44
1.2	30	25.00	25	0.00
1.12	30	26.79	25	7.14
1.03	30	29.13	30	-2.91
1.05	30	28.57	30	-4.76
1.1	30	27.27	30	-9.09
0.98	30	30.61	30	2.04
			Average % Error:	0.31

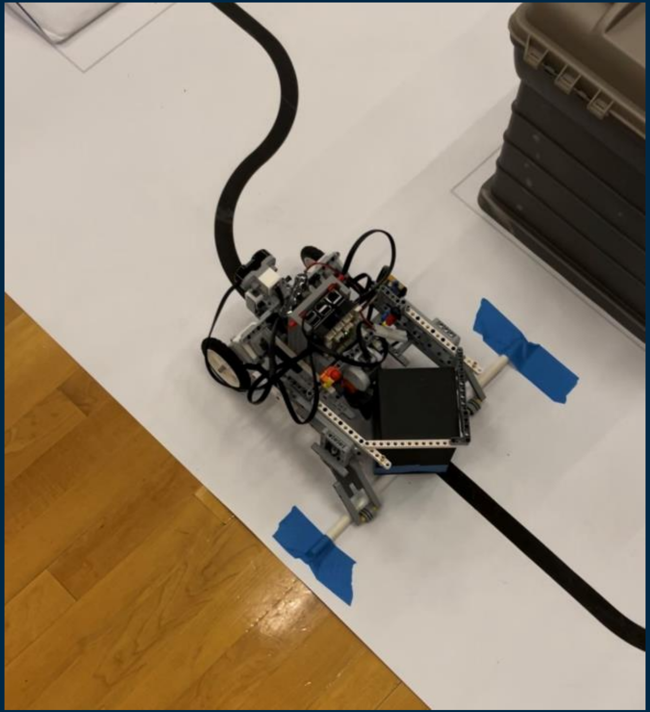
Negatives

- Inconsistency in scaling small obstacles
- Tuned for large obstacles, and struggles on smaller ones
- Small inconsistency in detecting magnetic markers
- Average test distance from designated drop off zone: 18cm



Areas for Improvement

- Consistency in scaling small obstacles
- Consistency in detecting magnetic markers
- Consistency in making accurate turns after magnetic markers



Thank You

Any Questions?