i) a code specification, as you discussed/developed with the TAs at the Prep meetings: an API
and a simple flow chart of the final program. (Please check with the TAs if unclear.)

<u>API & Simple Flow Char</u>

Sense.py

Low Level

class IR(io,pin):
__init__(self, io, pin)
read(self) → int
#Returns 1 (on tape) or 0 (off tape)

Mid Level

LineSensor(io)
__init__(self, io, pin_left, pin_middle, pin_right):
read() → return tuple[int, int, int]

DriveSystem.py

Low Level

Class Motor:
__init__(io, pin_LEGA, pin_LEGB)
stop()
setLevel(self, level)

Mid Level

Class DriveSystem:
__init__(self, io, left_pins ,right_pins)
drive(self, mode: str)
stop(self)

Top Level

Functionalities.py
(a) Flower-power.py
(b)
(c) IR_Testing.py
(d) Line-Following.py

<u>(ii) documentation if anything changed.</u>

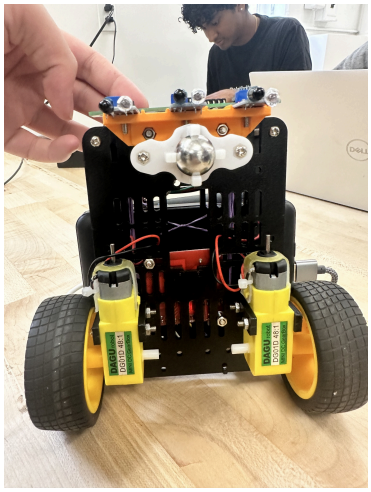
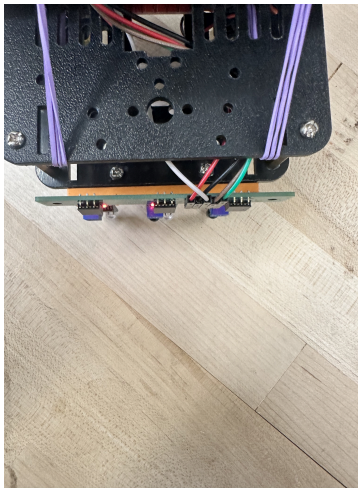
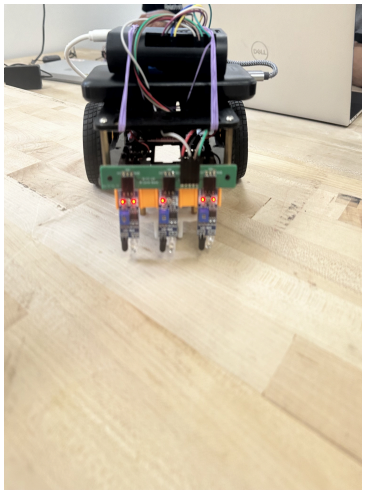
Updated Motor Class to correct mistakes from last week. Stop() Method is changed to self.setLevel(0) instead of directly setting the PWM to 0 to stop within the method. setLevel accounts for greater or equal to 0 instead of just greater than 0 for the if condition to prevent errors from occurring.

(iii) a table of PWM values for all options.

pwm_value = int(abs(level) * 250)

Movement	Left Motor	Right Motor
straight	0.75	0.75
Veer right	0.81	0.69
Steer Right	0.88	0.68
Turn right	0.945	0.585
Hook right	0.78	0
Spin right	0.75	-0.75
Veer left	0.71	0.80
Steer left	0.70	0.86
Turn Left	0.585	0.945
Hook left	0	0.78
Spin left	-0.75	0.75

iv) a photo of the IR sensors mounted.

v) the Feedback Law: a table of sensor readings to movements.

Case/Meaning	(L,M,R) Sensor Readings	Action
Centered	(OFF, ON, OFF)	STRAIGHT
Slight Left	(OFF, ON, ON)	STEER RIGHT
Far Left	(OFF, OFF, ON)	TURN RIGHT
SLIGHT RIGHT	(ON, ON, OFF)	STEER LEFT
FAR RIGHT	(ON, OFF, OFF)	TURN LEFT
OFF THE LINE	(OFF,OFF,OFF)	SPIN
SENSOR COVERING BLACK LINE	(ON,ON,ON)	STRAIGHT
EDGE CASE	(ON, OFF, ON)	SPIN