Summary

In this project, we created a remote-control robot with optional autonomous functionality. The robot incorporates 6 laser cut sides with holes for wiring and motors. Two dual-motor L298N h-bridge modules are used to control the four wheels. Three ultrasonic sensors are attached to the front of the robot for distance sensing. The device incorporates two embedded controllers: a khadas VIM3 board and an arduino mega. The Khadas VIM3 runs Ubuntu server 18.0.0.4. It serves a website using a program we wrote in c++. This program uses a library called oat++. Oat++ is used for websocket communications and API endpoints. Note that the video stream module code is taken from the Oat++ video over websockets demo (licensed under apache 2) and modified. The static site served by the program renders the video stream from the device, shows a distance telemetry overlay, and allows users to control the robot with key commands. This board treats the arduino as a peripheral device and sends commands over serial. The arduino interprets the commands and takes the specified action. The arduino board also controls autonomous mode. The ultrasonic sensors and h-bridge controls are plugged into the arduino board. Each H-bridge has a motor input from an individual battery back.

Arduino Code

The arduino code makes use of a simple state machine to interpret commands from the khadas board.

Command Set

Send: 0x00: A 0x01: ctr_byte_0 0x02: ctrl_byte_1	Sets motor powers to left = ctrl_byte_0, right = ctrl_byte_2.
Send: 0x00: B RECV: 0x01: Mode	Toggles autonomous vs manual mode. Sends T if in manual F if in autonomous
Send 0x00: T RECV: sensor_1:sensor_2:sensor_3	Gets telemetry from the three ultrasonic sensors, flushes the serial line and then sends the results.

State Transition Chart

State	Desc.	Next State
START	Checks serial input for state transition	SET_POWER_1 if Serial.read() == ‘A’, TOGGLE_1 if Serial.read == ‘B’. TELEM_1 if Serial.read == ‘C’
SET_POWER_1	Loads byte from serial into	If serial data,

	left motor power buffer	SET_POWER_2
SET_POWER_2	Loads byte from serial into right motor power buffer	If Serial data, START
TOGGLE1	Send T if in manual, F if in autonomous then Invert mode.	START
TELEM_1	Send sensor data on serial	START

Autonomous Algorithm

The arduino uses a simple algorithm using the three front autonomous sensors to navigate its environment. First, it checks to see if any of the sensor distances are less than 30cm. If so, it reverses the robot for one second. Next, it checks to see if the right ultrasonic sensor reading is less than the left ultrasonic reading. If this is true, it rotates left. Otherwise, it rotates right. If none of the sensor readings are less than 30, the robot moves forwards. If the left sensor is less than the right sensor, it adjusts its trajectory to the right. If the right sensor is less than the left sensor, it adjusts its trajectory to the left. Otherwise, it moves in a straight trajectory.

API endpoints

GET /telemetry	Returns telemetry data
GET /control?L_POWER=${l_power}?R_POWER=${r_power}	Sets left and right motor power
GET /v0/cam/stream	Returns static web page
WEBSCOKET /v0/cam	Returns stream from camers

The static webpage was written using HTML + CSS + js. The oat++ websocket stream demo code is used to render the video on an html canvas. A telemetry canvas is overlaid on top of the video canvas. The telemetry canvas draws on arc from pi radians to pi/2 radians. The arc is split into 3 areas with 3 segments per area. The first represented the left sensor and is drawn from pi radians to 1.33 * pi radians. The second represents the middle sensor and goes from 1.33*pi radians to 1.66 * pi radians. The third represents the right sensor and goes from 1.66*pi radians to 2 * pi radians. The three sectors represent three distance levels: < 20 cm: red, < 50 CM: yellow, < 100cm green. The site pulls data from the /telemetry endpoint and renders it on the canvas.

Changes from original plan

1. We removed the payload requirement. Incorporating payload carry with the current wire routing methods is not feasible.
2. We scaled from 6 ultrasonic sensors to 3. This works fine in autonomous mode and prevents wiring from getting even messier.

Moving forwards

1. Develop a more permanent wiring solution.

2. Migrate away from the L298N h-bridge modules. They are inefficient and lose a lot of energy to heat.
3. Use video encoder to transmit data faster.
4. Switch telemetry and robot control endpoints to websockets to reduce latency.
5. Higher torque motors are needed. The motors do not preform well when the robot is fully loaded: battery, camera, etc.