Car Hardware Connections

There will be 3 cars in the lab (labeled A, B, and C). If you need to remove a car from the lab, please let me know. Try to bring it back in a short time, so that others can have access as well.

The cars have the wheels removed for now. This makes testing a lot easier at first. I can give you the wheels upon request.

For the autonomous mode, I would recommend making it as simple as possible, at least at first. For example, you can have the car make a turn any time it is outside of a certain radius from its starting point. That way it should be able to wander around in a fashion similar to a Roomba robotic vacuum. Once the control scheme seems to be working, we can put the wheels on and test the system outdoors.

Remote Control (Transmitter) Arduino

Signal	**Arduino UNO pin**
Joystick button	10
x (steering) potentiometer	A2
y (throttle) potentiometer	A3

RFM69HCW RF Transceiver (recommended frequencies: car A 905MHz, car B 910MHz, car C 915MHz)

VIN	5V
GND	GND
G0	2
SCK	13
MISO	12
MOSI	11
CS	5
RST	6

OLED SSD1306 128x64 (uses I^{2}C for serial communication, address `0x3C`)

GND	GND
VDD	5V
SDA	A4
SCL	A5

On the remote control Arduino, the joystick is connected to pins A2 and A3. The joystick consists of two potentiometers acting as voltage dividers, so the input voltage signal is 0V to 5V. The joystick has a self-centering spring, so that if it is not pushed in any direction, both x and y signals return to the default 2.5V signal. The joystick incorporates a button; pressing the stick down connects digital pin 10 on the Arduino to ground.

If you like, you can use the OLED display to provide feedback to the user (mode, for example).

Receiver Arduino (on car)

Signal	**Arduino MEGA pin**
RFM69HCW RF Transceiver (recommended frequencies: car A 905MHz, car B 910MHz, car C 915MHz)	
VIN	5V
GND	GND
G0	18
SCK	13
MISO	12
MOSI	11
CS	7
RST	8

MinIMU-9 Inertial Measurement Unit (IMU) (uses I^{2}C for serial communication, accel & gyro uses address `0x6B`, magnetometer uses address `0x1E`, but those addresses are included in the `LIS3MDL.h` and `LSM6.h` libraries).

GND	GND
VDD	5V
SDA	20
SCL	21

Adafruit Ultimate GPS Receiver MTK3333 on a shield

VIN	5V
GND	GND
TX	10 (Arduino's RX pin)
RX	9 (Arduino's TX pin)

Steering servo control pin	4
Steering servo power supply	VIN
Steering servo ground	GND

Motor power supply	VIN
Motor ground	GND (by way of two npn BJTs)
Motor R encoder A pin	2 (logical interrupt 0)
Motor L encoder A pin	3 (logical interrupt 1)
Motor R control pin (to BJT base)	5
Motor L control pin (to BJT base)	6

The steering hobby servomotor is connected using VIN, GND, and Signal (digital pin 4).

The rear-wheel driving motors are connected to VIN and GND by way of two non bipolar junction transistors. Their speeds are controlled by pins 5 and 6, which supply current (through resistors) to the base of the two BJTs.

Only the encoder A pins are connected for each motor; the encoder B wires are left disconnected. This is because there are not enough interrupt pins. You can still use the encoder with one channel, but this cannot resolve direction. Use a 2x decoder scheme:

advance the counter every time Channel A rises or falls. If you added an H-bridge circuit, you could drive the motors in reverse, and monitor the speed.

NOTE: You don't need to use all of these devices! Use what you need. If you want to add other sensors, either make them easily removable or ask me if you want to mount something on the car long-term.

The remote and the car both have 9V rechargeable batteries to power them. Disconnect the batteries when not in use. Battery chargers are available with the cars.

NOTE: The RadioHead library uses a timer on the Arduino that is also used by the Servo library. To avoid this conflict, you can create the Servo control signal from scratch using `delayMicroseconds`. It isn't difficult.

You could also have the RadioHead library use a different timer. In this file:
Documents > Arduino > libraries > RadioHead > RH_ASK.cpp
Uncomment this line:

```
#define RH_ASK_ARDUINO_USE_TIMER2
```

Alternatively, you could keep the RadioHead on timer1, and switch the Servos to timer2, by using the `ServoTimer2` library:
https://create.arduino.cc/projecthub/ashraf_minhaj/how-to-use-servotimer2-library-simple-explain-servo-sweep-512fd9

NOTE: It seems to make the RF transceiver more reliable if you send nonzero bytes; it might interpret a zero byte as the end of the packet.

NOTE: Using hardware `Serial` connection to talk to the GPS receiver, instead of `SoftwareSerial` might help avoid problems with `SoftwareSerial` interfering with other libraries and functions. To make this switch, the GPS serial communication pins are shifted from Arduino pins 7 and 8 to pins 0 and 1.
You must hard wire pin 7 to pin 1, and pin 8 to pin 0.

```
#include <Adafruit_GPS.h>
#define GPSSerial Serial
Adafruit_GPS myGPS(&GPSSerial);
```

The `myGPS.begin(9600)` command remains, and is used to listen to the GPS receiver, not for Serial Monitor. **You must disconnect the wires connected to pins 0 and 1 in order to upload a program via USB. The easiest way to do this is to just unplug the *top* breadboard shield, upload, then plug it back in. Please remove the jumper wires so they don't mess up the next group.**

NOTE: Another option, which still allows for use of the SerialMonitor, is to use the NeoSWSerial library instead of SoftwareSerial. It causes fewer conflicts.