Copy Cat Car

By

David Fry

Sangamitra Katamreddy

Upendarredy Mamidi

Raghunath Pasumarthi

Contents

Abstract

The Copy Cat Car is a toy car which takes the commands from the user and replays those commands when needed. In our project we created a car that is controlled by the commands sent through a Bluetooth android application.

Our Basic idea was to send the commands through the Bluetooth application to a microcontroller and its output is then sent to a motor controller which is an H-bridge circuit and this controls the speed and direction of motors in the car accordingly the H-bridge is interfaced in UART mode and the Bluetooth module on the receiver's side is interfaced with the microcontroller through different pins.

The application will remember the course that was previously driven and replay it on command. This facilitated the need to create an android application that sends the HEX commands to the microcontroller. Throughout the project we learned about the UART serial interface and the android Bluetooth RFComm API library. In order to program for the phone we also had to understand basic Model view controller techniques and socket programming as the Bluetooth connection operates similarly to the Berkley sockets API.

Motivation

Our project was motivated by a simple idea, that it would be interesting to design a toy car which can be controlled by something instead of a remote controller. That can record a series of inputs from the user and store them in to the memory and plays them back when the user requests. This lead us think about an alternate to the classic RF remote controller which can send commands, quickly and efficiantly, but that also leaves the user without any feedback or replayability.

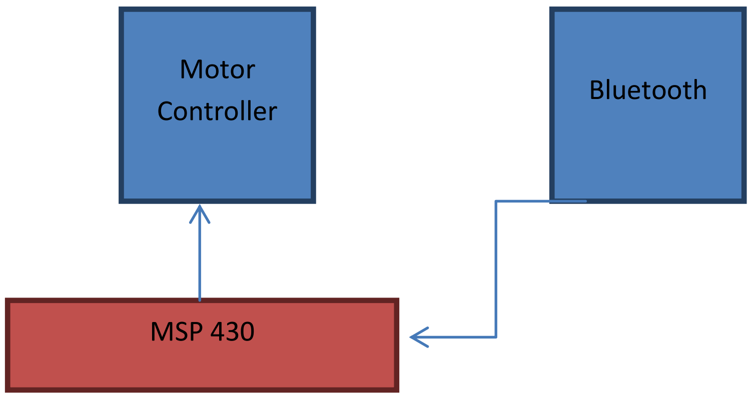
Bluetooth controlled devices are an explosive opportunity. Today many toys are being brought to market using bluetooth control and this made us to look to Bluetooth. We had a newfound desire to learn more about Bluetooth and the android operating system.

The second objective was to store the commands sent by the user and play them back when user needs. To achive this functionality the android application was written such that it was able to remember the sequence of commands given as well as the durration of those commands. By offloading this memory structure to the phone we were able to provide a much larger history.

The Bluetooth controller we would need to create is an Android application that would be able to create a bluetooth socket and send raw hexadecimal characters. The MSP430 would only be used as a buffer for each character. Originally we had planned on sending single byte characters through Bluetooth and interpreting them on the MSP before sending the intended commands to the motor controller but by using the android application to remember the route we found it to be much simpler to track the clock.

When implemented the program would take a difference in the linux epoch time codes and save that difference to an array with the speed of both motors. Upon pressing the replay button the application would then read through each line sending the motor commands and then sleeping for the value given by the time code.

Solution



Our solution is to use the msp430 as a buffer for the serial commands that will be sent directly through the Bluetooth adapter, this avoids the complexity of interpreting the commands as they are received.

Description

To begin with we developed the MSP code. This code was simplified from our original plan and only had to interface with the USART module to first setup the clock and enable the serial transmission mode. Then a reset signal is sent to the motor controller followed by a delay of 4 seconds to allow the motor controller to reboot. Once the motor controller is ready the system enables the art ax interrupt and goes into a deep sleep. Upon receipt of a Bluetooth command the MSP wakes and begins to transcribe the input serial bits to the transmit register. Once the interrupt routine has completed the MSP will return to a deep sleep.

The android application does most of the heavy lifting. When a user wants to drive the tank they set the speed with the two number picking wheels or the keyboard. Once settled upon a speed the user presses transmit and the application then reads the input numbers multiplies them by 3 to scale for the motor controller's inputs. These speed values are then added to a history array along with the Linux epoch time code the application then creates a serial data packet for each of the two motors.

This packet contains the motor controller's header of 0x8000 followed by the motor number either 3 for left or 5 for right and finally the motor speed in hex. These hex strings are then sent via the RFComm API interface to the Bluetooth module. To replay the previously input speeds the user presses a replay button, at that time the index for the history table is reset and the program begins to go through the history from the beginning while comparing the UTC values to determine the number of seconds between commands.

Results

Our results were to say the least disappointing. The commands which we are sending from the phone through the Bluetooth application are making the car move correctly but not for a significant amount time as the apparent action is that of a jump and not a driving motion.

All the components used were tested using an rs232 to usb adapter. They were found to be working perfectly even though they failed to operate when fully assembled. The android application was developed using eclipse and is also known to run correctly.

Using an off the shelf application and sending the commands that are typed in the command space of the application we were able to make sure that the application was not at fault. When any command is received by the motor controller the entire system crashes and will reboot. This leads us to believe that the power supply is actually at fault.

Appendix

Team members

David Fry – Android Application Programming, MSP430 Code, Hardware, Documentation

Sangamitra Katamreddy – Bluetooth interface,Slides and writing support

Upendarredy Mamidi –H-Bridge controller interface, Documentation and presentation slides

Raghunath Pasumarthi – Software and hardware interface , Android Application
Programming,MSP430Code.

Parts List

MSP430 Launchpad USB Programmer

MSP4302153 Microcontroller

2 .1uf Capacitors

2 3.3V DC Motors

Pololu Low-Voltage Dual Serial Motor Controller model no. 120

SparkFun RN41 Bluetooth Module

Tamiya Plastic dual Gearbox

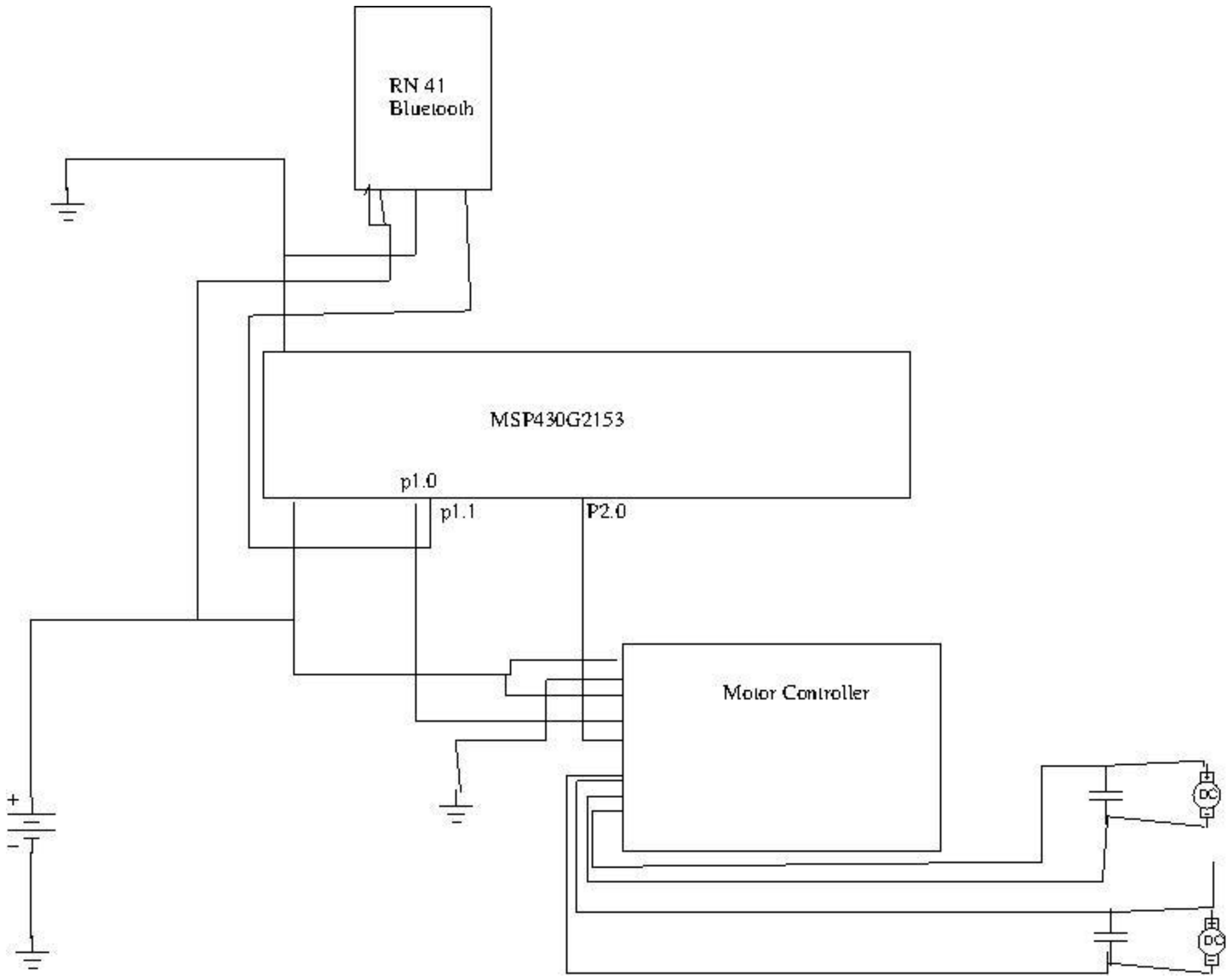
Tamiya Plastic Chassis and treads

4 AAA batteries

Radio shack 4 AAA battery holder

AnyVolt Voltage Regulator

Full Schematic



The Bluetooth module has an active low reset that must be tied into the +3.3v supply line. The Bluetooth also connects to the transmitter of the MSP430 on port 1.1. The motor controller has a reset that can be enabled by the port 2.0 and connects to the MSP430 through port 1.0 for serial data. The motors have .1uf capacitors connected in parallel to reduce noise created by the induction in the motor coils. The motor controller also has two separate power inputs one for the motors and one for the integrated circuits on the controller.