Authors: Mustafa Balcı and Tuğçe Çelt
Date: September 2, 2016

Introduction

This application note shows how to make a car/robot controller with Bluetooth and an Android app. It is all done using a SLG46620V GreenPAK4 CMIC, HC06 bluetooth module, motor drivers and motors. First, a simple android application was developed to send signals to the Bluetooth module, which in turn sends data to the SLG46620V. The motor is directed by processing the data with SLG46620V.

This project contains 3 stages :

1. Android Application
2. Retrieving data coming from the Bluetooth module
3. Controlling the motors

Android Application

In the first stage, we will compose an application for android devices to send data to the Bluetooth module. MIT app inventor 2 web site was used to compose the application. On this web site, you can easily create an application with programming blocks, without any programming knowledge. Meanwhile, let's think out what kind of data will be sent to the Bluetooth device before programming and designing.

Common directions a car basically moves in are left, right, forward and backward. To specify 4 different actions, 2 bits are enough (Table 1). After having specified car directional moves, let's set the speed of the car.

Because there are 4 different registers in the PWM0 block of the SLG46620V CMIC, we can adjust the speed to 4 different settings, and 2 binary bits are enough for this (Table 2).

Examples of direction:

D1	D0	Action
0	0	Go Back
0	1	Go Forward
1	0	Turn Left
1	1	Turn Right

Table 1. Bit Representation of direction

Examples of speed:

S1	S0	Speed
0	0	50 PWM
0	1	100 PWM
1	0	150 PWM
1	1	200 PWM

Table 2. Bit Representation of Speed

The data is sent to the Bluetooth module as 8 bit packages, and 5 bit data is enough for us. We can ignore the remaining 3 bits. We may align data as follows:

P7	P6	P5	P4	P3	P2	P1	P0
X	X	X	S1	S0	R	D1	D0

Table 3. Aligned Data

According to this bit sequence, the calculation of data that needs to be sent is as follows:

Action	Speed	8-Bit Represent	Decimal Represent
Go Back	50	XXX00100	4
	100	XXX01100	12
	150	XXX10100	20
	200	XXX11100	28
Go Forward	50	XXX00101	5
	100	XXX01101	13
	150	XXX10101	21
	200	XXX11101	29
Turn Left	50	XXX00110	6
	100	XXX01110	14
	150	XXX10110	22
	200	XXX11110	30
Turn Right	50	XXX00111	7
	100	XXX01111	15
	150	XXX10111	23
	200	XXX11111	31

Table 4. Data that is going to be sent to the Bluetooth device

Now we are able to create the application regarding these calculations.

4 buttons to define rotations, a list picker for devices and speed control, and a button for connection is enough. Also, we have to add bluetooth client and bluetooth server module to make a Bluetooth connection. To see directional buttons aligned, use the table arrangement. Design is shown in Figure 1.

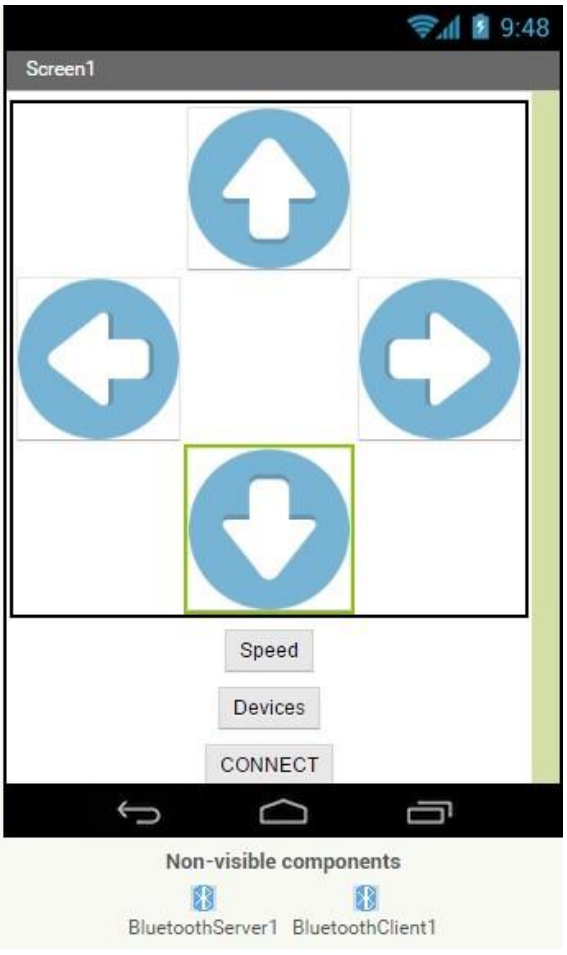

Figure 1. View of android app

In the first opening of application, remove the visible tick of remaining content one by one which is located in the properties section.

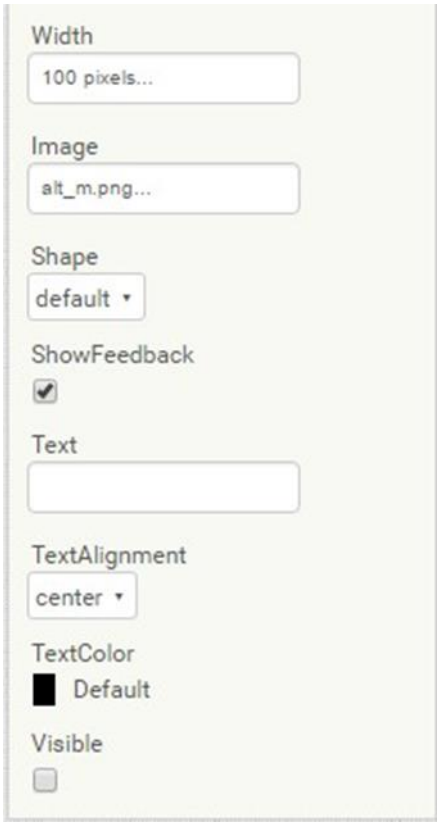

Figure 2. Properties of button

If you want to add figures instead of buttons, you can do this in image section.

After designing, we pass programming by clicking block section. Programming is quite easy with using Mix Application Inverter 2. You may add the block that you need by clicking components on the left side. In this part, we fill out the contents of the list pickers, and then line up the bluetooth devices so that the former match the devices, therefore, we arrange the numbers as ' 1, 2, 3, 4 ' for speed.

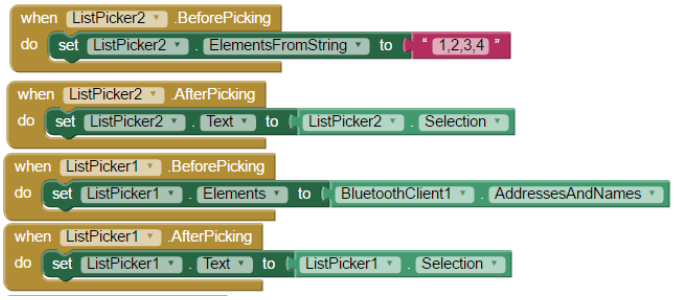

Figure 3. Programming blocks of list pickers

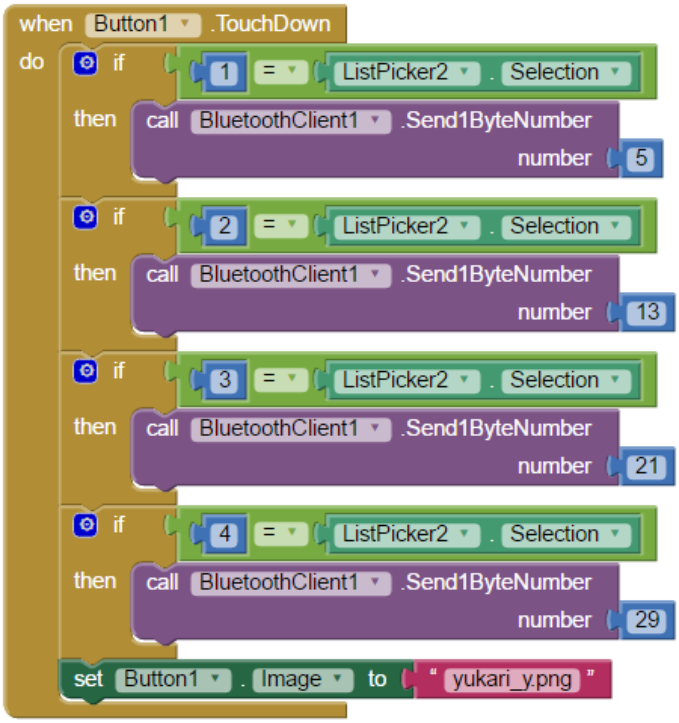

Figure 4. Programming blocks of go forward button

You can find the project of this Android Application in the files that you have downloaded for this project. If you select 'import project from my computer' by selecting project button located in the upper left, you can easily add your programs that you have made for your project.

Note: Some blocks are formed but are collapsed. You can right click on them and investigate by clicking expand button.

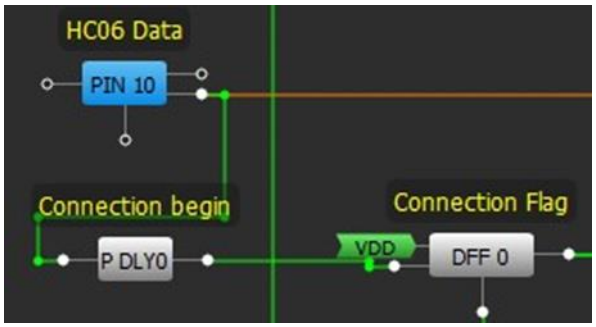

Figure 5. Start Bit Detector and Connection Flag

Getting Data from Bluetooth Module

HC06 module that we use for Bluetooth communication uses UART for communication protocol.

Bit number	1	2	3	4	5	6	7	8	9	10	11	12
	Start bit	5–9 data bits									Stop bit(s)	
	Start	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	Stop	

Table 5. Data frame of UART

A UART or Universal Asynchronous Receiver / Transmitter is a piece of computer hardware that translates data between parallel and serial forms. It is an integrated circuit used for serial communication that contains a receiver (serial to parallel converter) and transmitter (parallel to serial converter) each clocked separately.

Data Framing

The idle, no data state is high-voltage or powered. Each character is sent as logic low start bit, a configurable number of data bits and one or more logic stop bits.

HC06 sends 1 start bit, 8 data bit and 1 stop bit by default and also it has 9600 baud rate. SLG46620V has SPI block. We are going to make some additions to the SPI blocks which allows getting data that comes from HC06 module. Initially, an external clock is needed for SPI protocol. However, data comes at a constant rate under UART protocol. Another difference is that there are no start or stop bits in SPI communication. We are going to use a start bit to get SPI clock going.

UART protocol is in high power state when idle, and it passes to low power state when it starts communication (start bit - low).

Therefore, we can use Falling Edge Detector to identify the start of communication. We connected this signal to the clock pin of a DFF and VDD to data pin for the connection to stay active during the connection.

Thus, PDLY0 block is going to set connection flag by catching the falling edge when communication starts. UART protocol has 9600 baud rate which means it is able to transfer 9600 bits per second. To find the period of one cycle, you have to divide 1 second by 9600.

If you start the clock running immediately, data and clock signals will be changed at the same time and it may cause data loss. That is why clock signal must be delayed for a half cycle. You can better understand the necessity of this by referring to Figure 6.

We applied this delay using CNT6/DLY6 block. We attained a higher frequency by setting the RC OSC Frequency from the OSC block as 2000 kHz. The cycles from which we attain higher frequencies have shorter periods, and the absolute timing resolution of the counter increases. Thus, we decrease the counter's error ratio.

Note: The counters in the SLG46620V chip have +/- 1 error ratio. While the error ratio is 0,04 ms at 25 kHz, it is decreased to 0,0005 ms at 2000 kHz.

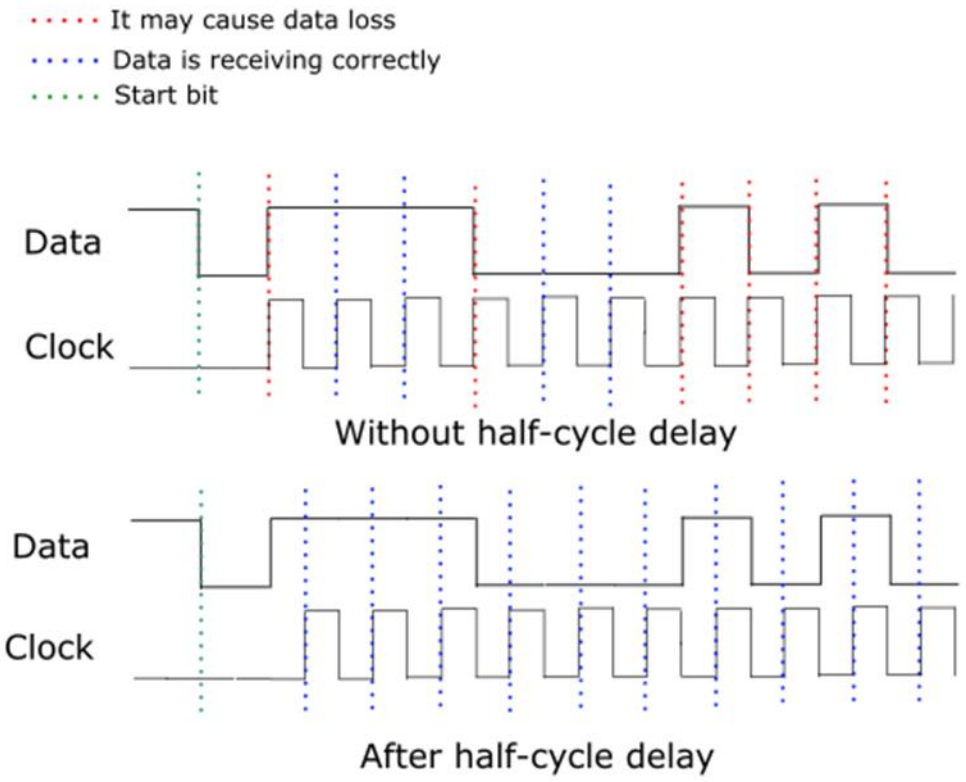

Figure 6. Timing (Before and after half-cycle delay)

CNT6 block is set in a way that it delays half a cycle (0,1042/2=0,0520 ms).

We used another counter (CNT2) to obtain a cycle's duration (0,142 ms). We set this counter's clock as External Clock0. By applying logic and process, we connected the OUT1 signal that comes from OSC and CLK BEGIN signal to OSC's External Clock0 input.

Thus, CNT2 will begin counting only when the connection starts. To give more detail, the counter blocks count the edges of the clock signals.

If the counter constantly sends high or low signals to the CLK pin, the counter stops counting. Here, we AND'd CLK Begin signal with OUT1 signal and connected them to the External CLK0 input. In this way, at the moment when there is no connection, ANDing CLK Begin signal and OUT1 signal will always equal to 0 and CNT2 will stop counting.

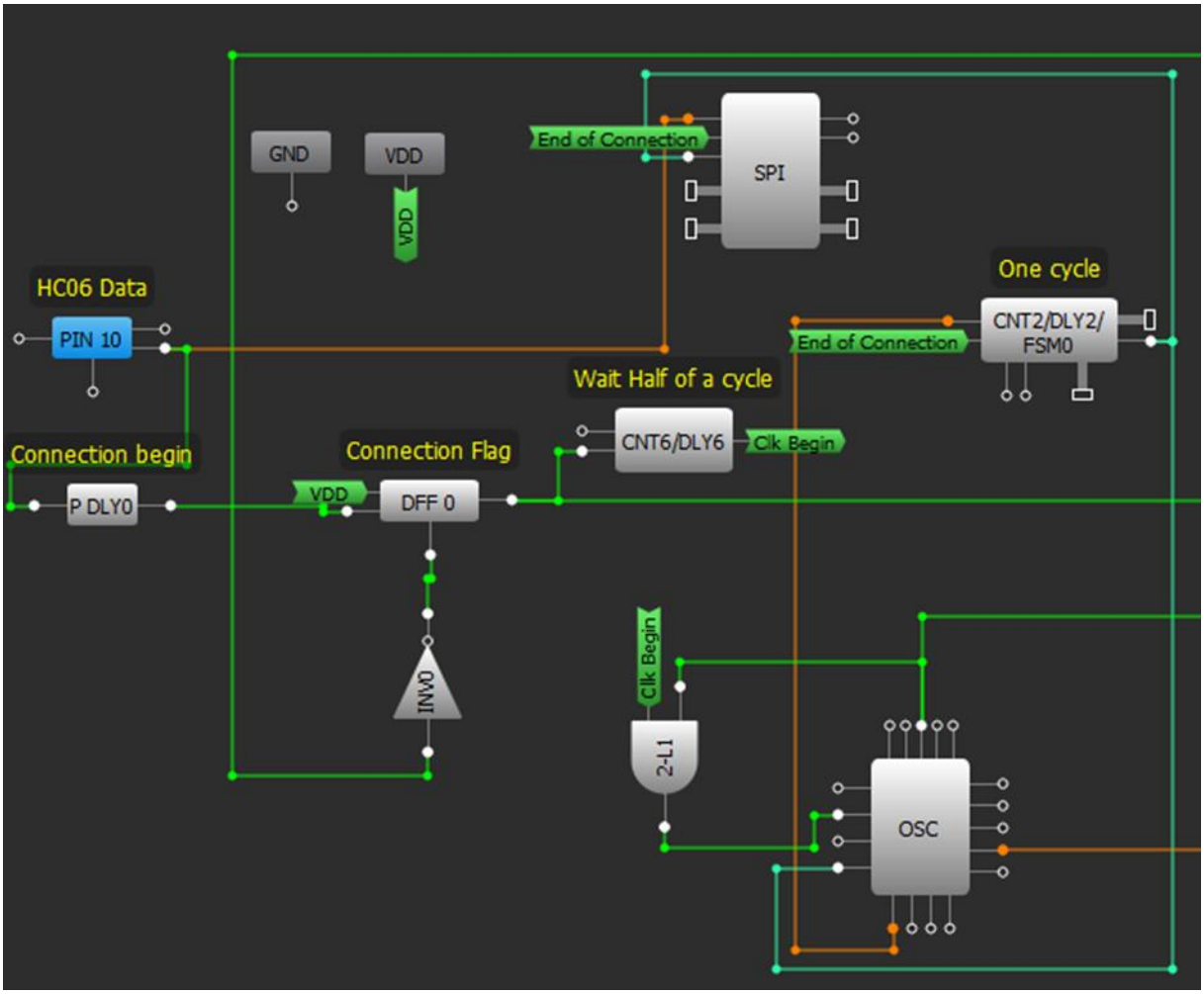

Figure 7. Connections of Counters

But when connection is provided, CLK Begin signal will be "1", the result of the logic and operation will be OUT1 signal and it will have CNT2 keep counting. We connected the output of the CNT2 counter (its cycle duration was calculated beforehand) to the SCLK pin of the SPI block. Hence, we generated a clock cycle for every byte of data coming from HC06.

Finally, we set a delay in order to calculate the connection's expiration and to zero the connection. This counter will begin measuring the duration as soon as the connection starts and will end the connection when the data delivery is completed. We know that the data coming from HCO6 is 10 bytes (1 byte start, 8 bytes data, 1 byte stop). We zeroed the connection after measuring the duration of 9 bytes. Zeroing the connection during stop byte makes the SPI block ready before a new connection is formed.

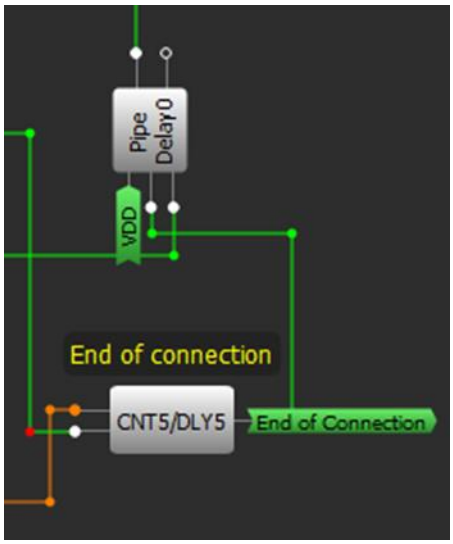

Figure 8. Applied delay to the end of the connection signal

Otherwise, if it is intended to be zeroed at the end of the duration of 10 bytes, there is the possibility of failure to capture the start byte for the new connection.

We reset the signal coming from the out pin of the PLY5 block, SPI block, the counter with 1 cycle duration we had calculated as well as DFF0 which we use as the connection flag.

However, unlike the other ones, we delayed the signal a little before resetting DFF0. If you reset the counter-delay blocks of the SLG46620V chip, the out chip of the block remains high for 1 cycle.

If you reset CNT2 and DFF0 at the same time, CLK Begin signal remains low, as a result of which stop CNT2's clock stops. As CNT2 will remain high for 1 cycle and the clock will shift to low status constantly, CNT2 will not be able to complete 1 cycle and will steadily remain in high status. Therefore, we reset the CNT2 block first and then DFF0 after a few cycles.

Control of the Motors/Car

At this stage, we change the car's speed and direction by changing the outputs of the DC motors. The car's direction and speed information is received via Bluetooth. As we receive the data coming through the UART protocol via SPI block, SPI Parallel Output block will be used here. Let's remember the Data Pattern which we previously used to control the vehicle.

D1	D0	Action
0	0	Go Back
0	1	Go Forward
1	0	Turn Left
1	1	Turn Right

S1	S0	Speed
0	0	50 PWM
0	1	100 PWM
1	0	150 PWM
1	1	200 PWM

Table 6. Speed and movement tables

PAR OUT 7	PAR OUT 6	PAR OUT 5	PAR OUT 4	PAR OUT 3	PAR OUT 2	PAR OUT 1	PAR OUT 0
D0	D1	R	S0	S1	-	-	-

Table 7. Reversed sequence

One of the differences between the UART and SPI communication is the sequence in which the bytes are sent. While MSB (Most Significant Bit) is sent initially with SPI protocol, it is sent last with UART protocol. Therefore, we will receive the bytes of the data with a reversed sequence.

The S1 and S0 bytes are the bytes by which we change the speed of the vehicle. So, we linked these bytes to the PWM0 block.

After the settings of PWM0 are applied as indicated above, the CNT8/DLY8 block is configured as indicated below.

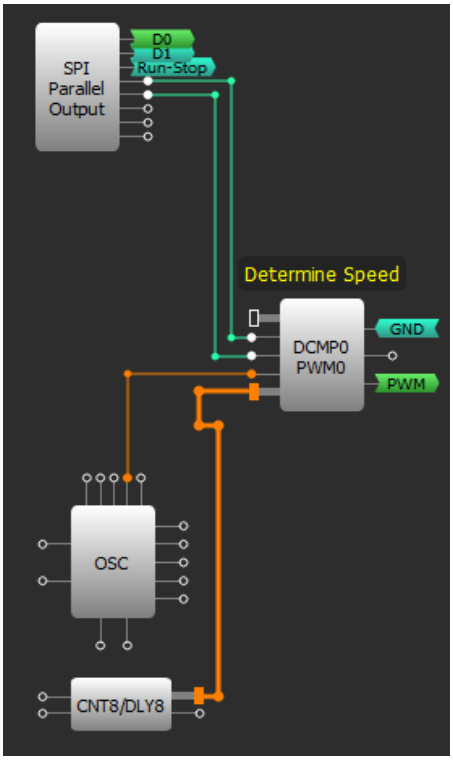

Figure 9. Connections of S1 S0 bits

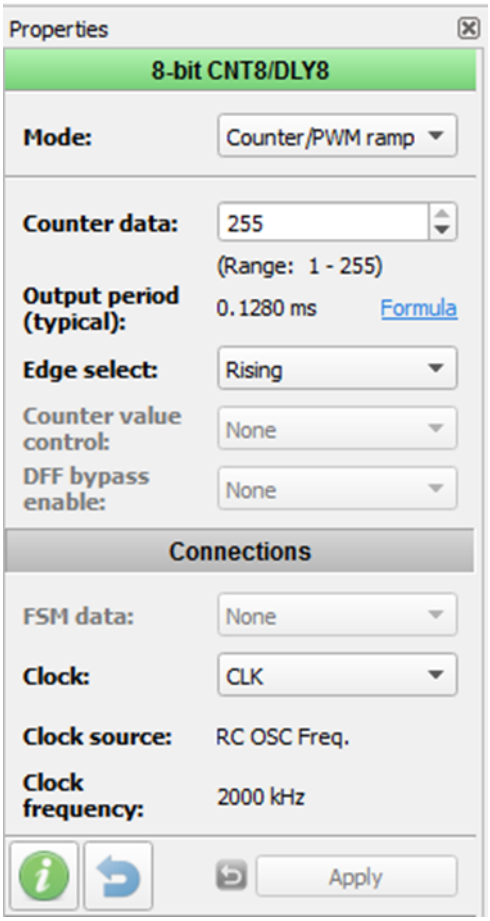

Figure 10. Settings of CNT8/DLY8

Now, we are able to receive PWM signal from the PWM0 block and adjust the signals necessary to set the car in motion. The SLG46620V CMIC cannot drive the DC motors directly. Therefore, we will be using a power supply and motor driver to control them. DC motor drivers run the motor by energizing the pins of the motor in accordance with the control pins that are usually named as M1-A, M1-B, M2-A and M2-B.

We build a connection in our design as shown below (Table 8).

M1-A	Left Motor (+)
M1-B	Left Motor (-)
M2-A	Right Motor (+)
M2-B	Right Motor (-)

Table 8. Motor driver's control pins

The signals that must be delivered to the DC motor control pins in accordance with the D1 and D0 signals are as shown in Table 9.

Action	D1 D0	M1-A	M1-B	M2-A	M2-B
Go Back	0 0	0	PWM	PWM	0
Go Forward	0 1	PWM	0	0	PWM
Turn Left	1 0	0	PWM	0	PWM
Turn Right	1 1	PWM	0	PWM	0

Table 9. Signals that are going to motor driver control pins

According to this table, the settings of the LUTs that control the motor out pins must be as displayed in Figure 11.

Finally, the LUTs supply the proper logic signals to the motor driver inputs. Thus, the motor out pins will remain low while the open-close signal is in the low state and the motors will not rotate.

3-bit LUT8					3-bit LUT9				
IN3	IN2	IN1	IN0	OUT	IN3	IN2	IN1	IN0	OUT
0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	1	1
0	0	1	0	0	0	0	1	0	0
0	0	1	1	1	0	0	1	1	0
0	1	0	0	0	0	1	0	0	0
0	1	0	1	0	0	1	0	1	1
0	1	1	0	0	0	1	1	0	0
0	1	1	1	1	0	1	1	1	0

3-bit LUT10					3-bit LUT11				
IN3	IN2	IN1	IN0	OUT	IN3	IN2	IN1	IN0	OUT
0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	1	0
0	0	1	0	0	0	0	1	0	0
0	0	1	1	0	0	0	1	1	1
0	1	0	0	0	0	1	0	0	0
0	1	0	1	0	0	1	0	1	1
0	1	1	0	0	0	1	1	0	0
0	1	1	1	1	0	1	1	1	0

Figure 11. Settings of LUTs

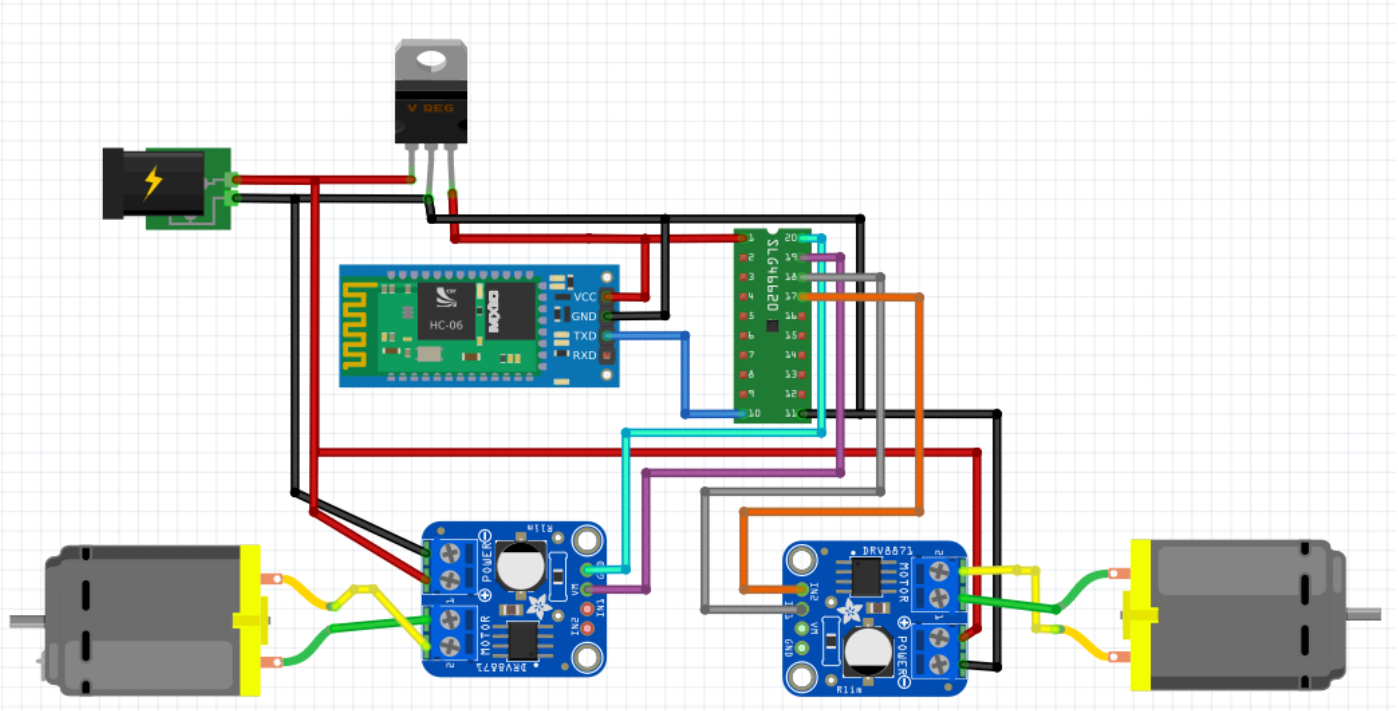

Figure 12. Top level schematic

Top Level Schematic

You can arrange the connections of the Bluetooth-Controlled Car as shown above in Figure 12. Since data only goes in one direction from the HC06 Bluetooth Module, we can just connect the Tx Pin. An engine drive card that can run two DC Motors can be used instead of 2 motor driver cards. Also, it would be useful to use a voltage regulator in order to protect Silego CMIC and HC06 Bluetooth Module from high voltage.

Conclusion

In this application note, control data was sent to the SLG46620V CMIC and operated wirelessly via Bluetooth. The Silego GreenPAK CMIC was successfully configured to support the UART protocol in data sequence and synchronization. Additionally, it can be synchronized with many sensors that use UART. This allows great flexibility designing with Silego GreenPAK CMIC for controls using smart devices (cell phones, tablets, etc.).

About the Authors

Name: Tuğçe Çelt

Background: Eskisehir Anadolu University

Student in the Electrical-Electronic engineering program

Name: Mustafa Balcı

Background: Eskisehir Anadolu University

Student in the Electrical-Electronic engineering program

Contact: **appnotes@silego.com**

Document History

Document Title: Bluetooth-Controlled Car/Robot

Document Number: AN-1120

Revision	Orig. of Change	Submission Date	Description of Change
A	Mustafa Balcı	09/02/2016	New application note

Worldwide Sales and Design Support

Silego Technology maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the sales person closest to you, visit us at **Sales Representatives and Distributors.**

About Silego Technology

Silego Technology, Inc. is a fabless semiconductor company headquartered in Santa Clara, California, with operations in Taiwan, and additional design/technology centers in China, Korea and Ukraine.

SILEGO TECHNOLOGY

Silego Technology Inc.
1515 Wyatt Drive
Santa Clara, CA 95054

Phone: 408-327-8800
Fax: 408-988-3800
Website: www.silego.com