BRICKS IN THE AIR

Interface Control Document

>> Background

This challenge was initially developed for Defcon27 in the Aviation Village. The concept was to create an environment that requires similar approaches to hacking on actual aviation buses without using any of the real hardware, protocols, or commands. Challengers can freely play and develop skills without worrying about legalities or sensitivities of real systems. This also makes it much cheaper and easier for people to replicate.

Common protocols used in the aerospace industry are MIL-STD-1553 and ARINC 429. While the protocols differ at the physical level they marshal data back and forth between connected devices in an implicitly trusted network. Gaining access to devices that communicate using these protocols is prohibitively expensive for those wanting to learn and often access is heavily regulated by industry stakeholders as they are reluctant to reveal how the protocols can be manipulated to cause unintended consequences.

To alleviate industry concerns and in striving to make an educational kit that has broad aerospace industry we chose instead to utilize the ubiquitous I2C protocol for this workshop. The I2C protocol has many similarities to both the MIL-STD-1553 and ARINC 429 protocols in that they all utilize the fundamental concept of Bus Controller and Receiver/Transmitter. Stated differently there is an orchestrating device (Bus Controller) that issues commands to listening devices (Receiver/Transmitter) that then in turn respond back with the desired information. Convention would state that only one device is to communicate at a time and each device compliantly waits to speak until spoken too. Within that construct, each device comes fully equipped to both listen and communicate. Therein lies the difficult problem to solve. How should these devices be safeguarded to ensure that communication flows smoothly as the engineers had intended. And what happens if that good order and discipline is disrupted?

High fidelity Lego Technic kits were chosen as the airplane stand-in because they provide a tangible mechanism to see, touch and feel what happens as a result of the information being distributed on the bus. That, and if we're being really honest, because Legos are just plain fun and we want to get as many people interested in that space as we possibly can. Each kit uses the Lego Power Functions motor and controller assemblies. In short, Lego Power Functions operate on 4 different IR channels with each motor connected to 1 of 2 connectors differentiated by the colors either Red or Blue. Complete documentation of the Lego Power Functions is available online here.

>> Overview

> DDS Board

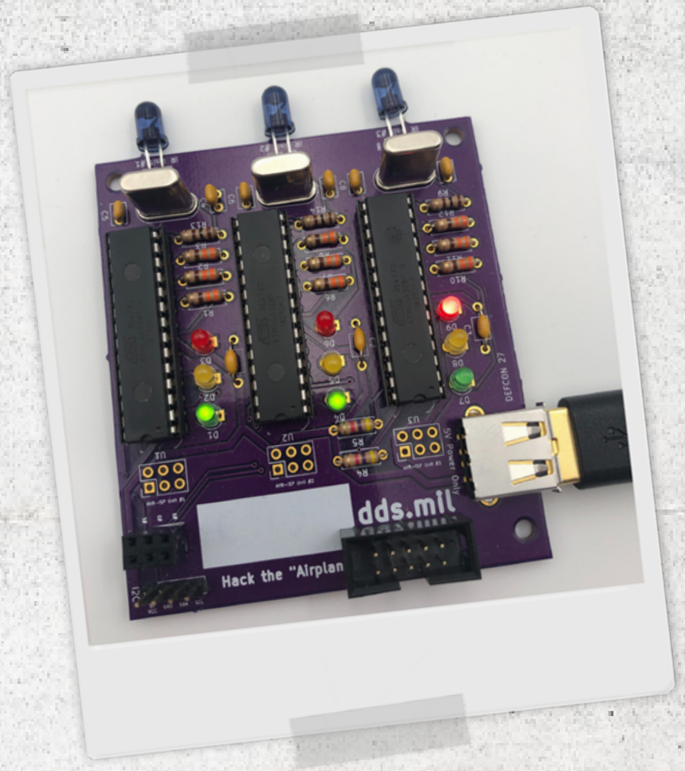
Each Lego kit is controlled through the native Lego Power Functions IR controller. DDS has designed and is using a custom PCB that has three onboard ATmega328 micro controllers. The Flight Control Computer is the Bus Controller and the other devices are receiver/transmitter on the same bus.

> Data Link

Protocol: I2C
Bus Speed: 100kbit/s
Logic Level: 5V
Hardware Pullups Present: True

The custom board is connected to a computer using an I2C to USB peripheral device that is controlled by a twitch chat bot written in Python, allowing users to interact by simply logging into a designated twitch stream.

Each of the three ATmega328 microcontrollers on the DDS board has a unique I2C address as outlined below and is able to communicate directly with a specific Lego Power Functions IR channel and color combination.



From left to right in the above image, the devices are:
Flight Control Computer - 0x50
Engine Control - 0x55
Gear Control/Other control - 0x60

>> Concept of Operations

> Flight Control Computer:
Often referred to simply as FCC: responsible for issuing and requesting the state of all connected devices. This device is responsible for the orderly communication of all connected devices. In normal modes of operation the FCC sends and receives information to all connected devices at regular intervals, thus ensuring that only it is the “allowed” communication device.

> Engine Control Computer:
Responsible for the safe operation of the main engines for the Lego kit as instructed by the Flight Control Computer. In normal mode of operation engine speed settings are limited to “safe” ranges.

> Landing Gear Computer:
Responsible for the timing sequence necessary to raise and lower the landing gear when instructed by the Flight Control Computer.

> Accessory Computer:
Responsible for the timing sequence necessary to drive a motor in a specific direction for a length of time. Examples include driving a winch up and down.

>> I2C Addressing

Title	I2C Address	Bus Con-troller	Receiver / Transmit-ter	Lego Power Function IR	LED Meaning
Flight Control Comput-er	0x50	X		X	Green - Normal Yellow - Secondary Mode of Operation Red - Maintenance Debug Mode
Engine Control Comput-er	0x55		X	X	Green - Normal Yellow - Secondary Mode of Operation Red - Maintenance Debug Mode
Landing Gear Comput-er	0x60		X	X	Green - Extended Yellow - In transit Red - Retracted

>> **General Information** Flow

Lego Kit
Lego Power Functions IR interface*
Custom DDS board that is using the ATmega328 MCU and programmed using Arduino
I2C*
I2C to USB Interface
Twitch Chatbot written in Python*
Users Browser

*indicates interface between components

>> Twitch Integration control

With integration into twitch.tv the following is a summary of how to interact with the kits.

! - leading character to indicate the following text needs to be processed

Text	Description
join	Join the active cue to interact with the kit. Note: you will be removed from the active cue after a certain period of inactivity.
leave	Leave the active cue and allow others to play.
cmd	Send an I2C command to the custom board to interact with the Lego Power Functions. If a correct command is sent that would result in the correct answer the user will be advanced to the next question.
question	Display the current question in the chat window
hint	Request a hint for the current question in the chat window.

>> Example Commands

Interactions	Description
!join	Join the active cue
!cmd 0x50 0x80	Send the I2C command to address 0x50 (FCC) send the request 0x80 (Get Lego PF Channel)
Hey! Wazzup? This is fire!	Chat with your buddies and tell them how cool the challenge is.
!hint	Get a hint for the current question that you're on.

Interface Control Documentation - Tables

Flight Control Computer 7 bit address: 0x50			
Command	Payload	Response	Description
Get Engine Speed 0x10	Size: NA	Size: 1 byte 0x00 - 0x07	Get the speed of the engine
Set Engine Speed 0x11	Size: 1 byte	Size: NA	Set the engine speed 0x00 - 0x07 Note: safety systems prevent turning the engines off in flight
Get Gear Position 0x20	Size: NA	Size: 1 byte 0x00 Extended 0x01 Retracted 0x02 In transit	Get the current state of the landing gear.
Set Gear Position 0x21	Size: 1 byte 0x00 Extend 0x01 Retract	Size: NA	Set the Landing Gear position
Get Mode of Operation 0x30	Size: NA	Size: 1 byte 0x00 Primary mode of operation 0x01 Secondary mode of operation	Get the current operational mode
Set Mode of Operation 0x31	Size: 1 byte 0x00 Primary Mode 0x01 Secondary Mode	Size: NA	Set the mode of operation
Get State 0x35	Size: NA	Size: [] bytes	Get the current operational state of the device and all connected devices
Get Maintenance Status 0x40	Size: NA	Size: 1 byte 0x00 Normal 0x01 Debug	Get the maintenance status
Set Maintenance Status 0x41	Size: 1 byte 0x00 Normal 0x01 Debug	Size: NA	Used for maintenance and troubleshooting inquiries.
Send Message to R/T 0x51	Size: 3 bytes 1 byte address 1 byte command 1 byte payload	Size: NA	Send a message directly to a R/T

Flight Control Computer 7 bit address: 0x50 (continued)			
Command	Payload	Response	Description
Get Lego PF Channel 0x80	Size: NA	Size: 1 byte 0x00 Channel 1 0x01 Channel 2 0x02 Channel 3 0x03 Channel 4 0x04 No Connection	Get the Lego PF this device is connected to.
Get Lego PF Color 0x90	Size: NA	Size: 1 byte 0x00 RED 0x01 BLUE 0x02 No Connection	Get the Lego PF color this device is connected to
Unknown	Size: NA	Size: 1 byte 0x33	Response when sent an invalid command

Engine Control Computer 7 bit address: 0x55			
Command	Payload	Response	Description
Get Engine Speed 0x10	Size: NA	Size: 1 byte 0x00 - 0x07 Speeds 0-7 0xDA Fault Detect-ed	Get the speed of the engine
Set Engine Speed 0x11	Size: 1 byte	Size: 1 byte 0x00 Rejected Change 0x01 Accepted Change 0xDA Fault Detect-ed	Set the engine speed 0x00 - 0x07 Note: safety sys-tems prevent turn-ing the engines off in flight
Get Mode of Opera-tion 0x30	Size: NA	Size: 1 byte 0x00 Primary mode of operation 0x01 Secondary mode of operation	Get the current operational mode
Set Mode of Opera-tion 0x31	Size: 1 byte 0x00 Primary Mode 0x01 Secondary Mode	Size: NA	Set the mode of operation
Get Maintenance Status 0x40	Size: NA	Size: 1 byte 0x00 Normal 0x01 Debug	Get the mainte-nance status
Set Maintenance Status 0x41	Size: 1 byte 0x00 Normal 0x01 Debug	Size: 1 byte 0x00 Rejected Change 0x01 Accepted Change	Used for mainte-nance and trouble-shooting inquiries.
Get Lego PF Chan-nel 0x80	Size: NA	Size: 1 byte 0x00 Channel 1 0x01 Channel 2 0x02 Channel 3 0x03 Channel 4 0x04 No Connection	Get the Lego PF this device is connected to.
Get Lego PF Color 0x90	Size: NA	Size: 1 byte 0x00 RED 0x01 BLUE 0x02 No Connection	Get the Lego PF color this device is connected to
Unknown	Size: NA	Size: 1 byte 0x33	Response when sent an invalid command

Landing Gear Computer 7 bit address: 0x60			
Command	Payload	Response	Description
Get Gear Position 0x20	Size: NA	Size: 1 byte 0x00 Extended 0x01 Retracted 0x02 In transit	Get the current state of the landing gear.
Set Gear Position 0x21	Size: 1 byte 0x00 Extend 0x01 Retract	Size: NA	Set the Landing Gear position
Get Mode of Operation 0x30	Size: NA	Size: 1 byte 0x00 Primary mode of operation 0x01 Secondary mode of operation	Get the current operational mode
Set Mode of Operation 0x31	Size: 1 byte 0x00 Primary Mode 0x01 Secondary Mode	Size: NA	Set the mode of operation
Get Maintenance Status 0x40	Size: NA	Size: 1 byte 0x00 Normal 0x01 Debug	Get the maintenance status
Set Maintenance Status 0x41	Size: 1 byte 0x00 Normal 0x01 Debug	Size: NA	Used for maintenance and troubleshooting inquiries.
Get Lego PF Channel 0x80	Size: NA	Size: 1 byte 0x00 Channel 1 0x01 Channel 2 0x02 Channel 3 0x03 Channel 4 0x04 No Connection	Get the Lego PF this device is connected to.
Get Lego PF Color 0x90	Size: NA	Size: 1 byte 0x00 RED 0x01 BLUE 0x02 No Connection	Get the Lego PF color this device is connected to
Unknown	Size: NA	Size: 1 byte 0x33	Response when sent an invalid command