BACKGROUND

A few years ago, we saw a picture in an engineering magazine of a man welding what appeared to be an elephant. We wrote to the magazine for more details and they told us it was a chap called Frank Stuart. Further research led to learning about his company Mechanimal who made a number of petrol-powered elephants in the 1950s. These ride on elephants could carry a number of children around a park or zoo. Frank's motivation was to cut the costs of keeping live elephants.

Back in the 1950s an estimated 250 elephants were being killed each day for ivory and meat. Which led to a world Ivory ban in the 1990s. This is now down to 100 a day but as this is more than the number of baby elephants being born, elephant numbers are still declining. World Elephant day on 12th August was created to spread the message about this problem and to educate people on how they can help

http://worldelephantday.org

THE BROAD OVERVIEW

This elephant model is moved by 4 servos controlled by a web server running on the ESP32 board. When the forward button is pressed the servos walk the elephant forward. When the left button is pressed, the legs on the left take smaller steps causing the elephant to steer to the left. When the right button is pressed it steers to the right.

PROTOTYPE

Not really having the space to create and run a full-sized elephant we decided on a model one instead.

We looked at a range of different walking robots from the very simple to the extremely sophisticated. The key features we wanted were that it could move on it own without external wires and could be steered with some form of remote control.

The first test rig was just 4 servos strapped together. This allowed walking algorithms to be tested and to see if the servos would have enough power to carry a battery.

Components
4 X Servo
1 X Arduino Uno

<IMAGE>TestRig.jpg</IMAGE>

```
/* Sweep
 *  Based on the work of BARRAGAN and Scott Fitzgerald
 *  http://www.arduino.cc/en/Tutorial/Sweep
*/

#include <Servo.h>

Servo myservo[4];

int position[4];
int direction[4];

int minpos = 50;
int maxpos = 120;
int slow = 1;
int fast = 4;

void setup() {
  // 3, 5, 6, 9
  myservo[0].attach(3);
  myservo[1].attach(5);
  myservo[2].attach(6);
  myservo[3].attach(9);

  position[0] = minpos;
  position[1] = maxpos;
  position[2] = minpos;
  position[3] = maxpos;
}

void moveServo(int servo, int speed) {
  myservo[servo].write(position[servo]);
  position[servo] = position[servo] + speed;
  if (position[servo] < minpos) { position[servo] = minpos; }
  if (position[servo] > maxpos) { position[servo] = maxpos; }
}

void loop() {
  for (int pos = minpos; pos <= maxpos; pos += 1) {
        moveServo(0,slow);
        moveServo(1,-fast);
        moveServo(2,-slow);
        moveServo(3,fast);
        delay(15);
    }
  for (int pos = minpos; pos <= maxpos; pos += 1) {
        moveServo(0,-fast);
        moveServo(1,slow);
        moveServo(2,fast);
        moveServo(3,-slow);
        delay(15);
    }

}
```

The test rig allowed us to see two things. Firstly that the simple back and forth motion of the servos would not cause the elephant to walk. Something else would be needed. I also allowed us to test the power requirements of the servos. At 5v they took around 300mA when they were all moving. They did move with a supply voltage down to 3v but the torques as greatly reduced and the servos were easily stalled.

3D MODEL

We designed the Elephant as a 3D model using Fusion 360.

The first part of the 3D model to be created was the servo. Given the shape and size of these, they will influence the scale and form of the elephant. A frame was designed to hold these, we added slots for the cables and screw holes for mounting the servos.

Next up was the leg. Frank's elephants had a wheel at the end of each leg. A simple forward backwards motion combined with a brake on the wheels propelled the elephant forward. Hydraulics kept the elephant level and ensured the wheels were always on the ground.

Based on the experiments we had done with the test rig, it was clear we would need to do something similar to ensure that the leg produced forward traction. The other key fact gleaned from the testing was that the servos on each side need to move in the opposite direction. So to move the left legs forward the servos would rotate clockwise and to move the right legs forward, anti-clockwise.

For our elephant, it did not matter if the elephant wobbled so the hydraulics were not needed, for the wheels, we added ridges around the outside and a pawl to form a ratchet. The servo operates the upper leg and a wire link raises the knee.

<Image RatchettMechanism.png>

The head was a complex part to model, the best tool to create the organic shape was the “form” tool as that uses Nurbs to produce a curved shape. These could be pulled and pushed to create the shape of the head. To create the trunk, a flat was sliced off and the resulting surface extruded along a curved path using the sweep tool. The ears were also created as a “form” and then subtracted from the head so they could be printed separately and slotted in.

The body needed to be hollow to house the electronics. So this was made as a form then sliced up and emptied out using the shell command.

Several test prints were made to check the parts fitted together and moved as intended.

Some of the components were flagged by the slicer as having invalid geometry. This usually occurs when two parts are very close but not joined. It can also occur when a part is too thin. If you can't fix these issue at source tools like MeshMixer can repair the mesh for you. Fusion360 also has an inbuilt patch tool for repairing meshes by hand.

THE MAIN BUILD

PARTS

1 x https://www.altronics.com.au/p/z6385-esp-32s-wifi-bluetooth-module-and-interface-board/#fancy-inner

4 x https://www.altronics.com.au/p/z6392-9g-180deg-plastic-servo-for-arduino/

1 x https://core-electronics.com.au/powerboost-500-basic-5v-usb-boost-at-500ma-from-1-8v.html

1 x https://www.altronics.com.au/p/s2010-dpdt-solder-tail-sub-miniature-slide-switch/

1 x 1000uF Electrolytic capacitor
https://www.jaycar.com.au/1000uf-25vdc-electrolytic-rb-capacitor/p/RE6230

1 x 220R resistor -
https://www.jaycar.com.au/220-ohm-1-watt-carbon-film-resistors-pack-of-2/p/RR2558

Screw terminals -
https://www.jaycar.com.au/2-way-pcb-mount-screw-terminals-5mm-pitch/p/HM3172

Headers - 0.1
https://core-electronics.com.au/header-40-pin-male-long-centered-pth-0-1.html

1 x Red LED 3mm
https://core-electronics.com.au/t1-3mm-red-led-with-red-diffused-lens.html

1m of
https://www.jaycar.com.au/mains-20a-twin-and-earth-power-cable-sold-per-metre/p/WB1568

4x M3 rod (brass or glass fibre) for the wheel axles.
4x M2 x 8mm bolt
https://www.jaycar.com.au/metric-metal-thread-fasteners-2x8mm/p/HP0390
8x M2 nut
4x G3 x 10mm self-tapping screw
4x G4 x 12mm self-tapping screw
https://www.jaycar.com.au/no-4-x-12mm-steel-self-tapping-screws-pk-25/p/HP0554
8x G2 x 6mm self-tapping screw

Alternative supplier for screws -
https://www.smallparts.com.au/store/partscombtab/selftappingscrewsforplastics/?v=8592

BREAKOUT BOARD

To provide power and act as a place to easily connect all the servos, a breakout board was created. The key design features were that it should have an LED so we can confirm power is enabled, screw terminals for the input power and standard 3 pin headers for the servos.

<IMAGE>Breakoutboard.jpg</IMAGE>

The assembly of this is straightforward, it will be easiest if you install the connectors first and then the larger components. Check the polarisation of the LED and Capacitor. You can test this breakout independently if the main build.

ELEPHANT BRAINS WITH ESP

The controller for our mini elephant robot is an ESP32 board. This is the newer version of the ESP2866 which comes with a faster dual-core processor and the usual WiFi connectivity.

To configure your computer for developing with the ESP32 you may need to install a virtual com port driver for the USB to serial chip. This will be either a CH340 or CP2102 depending on where you bought your board from.

https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers

Once the driver installed you should find that the board appears as a comm port when you plug it into your computer.

You'll also need to configure the Arduino IDE to talk to this board. From the Arduino IDE, select the File menu, then Preferences. In the “Additional Boards Manager URLs:” box type add the following.

https://dl.espressif.com/dl/package_esp32_index.json

Then from the Boards Manager install “ESP32 by Espressif Systems”

The ESP32 can produce the PWM signals for the servos on any of the available pins. It can support upto 16 channels at once. It does need a different library instead of the default servo library.

To install the library, choose the library manager, and search for ESP32Servo.

The elephant communicates to your phone or computer via a web page. The name resolution is done using a technology called mDNS or ZeroConfig. This allows us to access our board as ESP32.Local rather than having to know it’s IP address.

For this to work you may need to install some software.

- For Linux, install Avahi (http://avahi.org/).
- For Windows, install Bonjour (http://www.apple.com/support/bonjour/).
- For Mac OSX and iOS support is built in through Bonjour already.

CODE

For the final code we simplified the servo motion so that forward and backwards movement was at the same speed. To turn the elephant we reduced the stride length on the side it turns towards.

The code includes the libraries we need to connect to the wifi, resolve the name, run the webserver and move the servos.

```
#include <ESP32Servo.h>
#include <WiFi.h>
#include <WiFiClient.h>
#include <WebServer.h>
#include <ESPmDNS.h>
#include "Secrets.h"
```

This is followed some special constant. These “SECRET” constants work with the Secrets.h file and if you use the webIDE these will be kept securely. For the desktop Arduino IDE, you have to remember not to copy the secrets file to your source control system.

```
const char *ssid = SECRET_SSID;
const char *password = SECRET_PASS;
```

The next constant is the definition of the main page shown by the webserver. This has been minified and converted to a string using the following two tools. The full code for this page is provided in the resources. It presents to the users some buttons which when pressed make REST calls back to the webserver with the commands, left, right, forward or stop. This variable is stored in program memory using the PROGMEM keyword.

The webserver is declared and defined to run on the standard port 80.

<image web page.png>

https://www.willpeavy.com/tools/minifier/
http://tomeko.net/online_tools/cpp_text_escape.php?lang=en

```
const char PROGMEM *webpage = "<!DOCTYPE html><html><head><meta
charset=\"UTF-8\"><title>Elephant Control Panel</title><script>function
move(dir){document.getElementById(\"response\").innerHTML=dir; var xhr=new
XMLHttpRequest(); xhr.timeout=500; xhr.ontimeout=function
(e){document.getElementById(\"response\").innerHTML=\"Elephant not responding\";};
xhr.onreadystatechange=function(){if
(this.readyState==4){document.getElementById(\"response\").innerHTML=this.responseText
;}}; xhr.open(\"POST\", \"/\" + dir, true); xhr.send();}</script></head><body> <nav> <p>
<button type=\"button\" onclick=\"move('left')\">&lt;</button> <button type=\"button\"
onclick=\"move('forward')\">^</button> <button type=\"button\"
onclick=\"move('right')\">&gt;</button> </p><p><button type=\"button\"
onclick=\"move('stop')\">Stop</button></p></nav> <p id=\"response\"></p></body></html>";

WebServer server(80);

Servo myservo[4];
int strideLeft;
int strideRight;

const int strideShort = 20;
const int strideLong = 45;
const int strideStopped = 0;
const int middlePosition = 90;
```

We also declare the array containing our 4 servos and define some variable which will be used to control the motion of the legs.

In the setup, the first tasks are to connect to the Wifi and define the name in MDNS as “esp32”. We then define what happens when the user requests different web pages. After starting the webserver we configure the servos to connect to the different pins of the ESP32 board. The servos are moved to their starting positions.

```
void setup(void) {
```

```
Serial.begin(115200);
WiFi.mode(WIFI_STA);
WiFi.begin(ssid, password);
Serial.println("");

// Wait for connection
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}
Serial.println("");
Serial.print("Connected to ");
Serial.println(ssid);
Serial.print("IP address: ");
Serial.println(WiFi.localIP());

if (MDNS.begin("esp32")) {
  Serial.println("MDNS responder started");
}

server.on("/", []() {
  server.send(200, "text/html", webpage);
});

server.on("/left", []() {
  strideLeft = strideShort;
  strideRight = strideLong;
  server.send(200, "text/plain", "Turning Left");
});

server.on("/right", []() {
  strideLeft = strideLong;
  strideRight = strideShort;
  server.send(200, "text/plain", "Turning Right");
});

server.on("/forward", []() {
  strideLeft = strideLong;
  strideRight = strideLong;
  server.send(200, "text/plain", "Straight on");
});

server.on("/stop", []() {
  strideLeft = strideStopped;
  strideRight = strideStopped;
  server.send(200, "text/plain", "Stopping");
  });

  server.begin();
  Serial.println("HTTP server started");

  myservo[0].attach(25); //D2
  myservo[1].attach(26); //D3
  myservo[2].attach(27); //D4
  myservo[3].attach(9); //D5

  myservo[0].write(middlePosition);
  myservo[1].write(middlePosition);
  myservo[2].write(middlePosition);
  myservo[3].write(middlePosition);

  strideLeft = strideStopped;
  strideRight = strideStopped;
}
```

In the main loop we handle the requests for the webserver and the motion of the legs.

```
void loop(void) {
  server.handleClient();

  if (strideLeft != strideStopped) {
  //Front right and rear left legs forward
    for (int pos = 0; pos <= 45; pos += 1) {
          if (pos < strideRight) { myservo[0].write(middlePosition + pos); }
          if (pos < strideLeft)  { myservo[2].write(middlePosition - pos); }
          delay(15);
      }
    //Front left and rear right forward
    for (int pos = 0; pos <= 45; pos += 1) {
          if (pos < strideRight) { myservo[1].write(middlePosition + pos); }
          if (pos < strideLeft)  { myservo[3].write(middlePosition - pos); }
          delay(15);
      }
    //Front right and rear left backwards
    for (int pos = 45; pos > 0; pos -= 1) {
          if (pos < strideRight) { myservo[0].write(middlePosition + pos); }
          if (pos < strideLeft)  { myservo[2].write(middlePosition - pos); }
          delay(15);
      }
    //Front left and rear right backwards
    for (int pos = 45; pos > 0; pos -= 1) {
          if (pos < strideRight) { myservo[1].write(middlePosition + pos); }
```

```
        if (pos < strideLeft)  { myservo[3].write(middlePosition - pos); }
        delay(15);
    }
  }
}
```

CONSTRUCTION

The parts for the elephant were printed in a Silver Grey ABS from Rigid Ink. Including the test prints we used about 40m of filament.

The parts have been designed with one flat side and manageable overhangs and spanning. Orientate the prints with the flat side down. For the upper body, you may find it helps to slow the print speed as it gets near the top. The small size of the pawl will also benefit from a slower print rate.

The head, ears and tail will likely need supports.

Parts

Part	Quantity
Upper body	1
Middle body	1
Lower body	1
Head	1
Left ear	1
Right ear	1
Upper leg	4
Left lower leg	2
Right lower leg	2
Wheel	4
Pawl	4
Tail	1

SERVO CALIBRATION

Before assembling the elephant you will need to know the positions of the servos. We did this with a simple sketch that set all the servo positions to 90 degrees. You could also do this with the servo tester from Diyode #24.

ASSEMBLY

Before assembly ensure all the parts are deburred paying particular attention to the holes for screws and spindles.

The lower body should be glued onto the middle body, flat side to flat side. The upper body is designed to clip on and off so you can get at the electronics.

The ears should glue onto the head and the head and neck glue to the body. You may need to support the head whilst the adhesive sets.

Remove the horns from the servos. Fit the servos into the chassis so that the spindles are closest to the ends. Take care to ensure the cables don't get crushed and that they all exit from the same side. Screw the servos in place using the tiny G2 screws.

<Image servos in chassis>

It may be worth labelling the servo cables at this point. Feed the servo wires through the holes in the lower body. The front right leg is 0, rear right 1, rear left 2 and front left 3.

The lower body should be a push fit onto the chassis.

Assemble each of the legs by fitting the G4 screw through the knee. The wheel slots into the bottom of the leg, check the orientation of this as the sloping side of the ratchet should face up. Slot in an axle and check that the wheel rotates freely. If not then remove the axle and use a drill or file to open up the hole in the wheel slightly before reassembly. The pawl bolts onto the lower hole in the leg using the M2 bolt and should allow the wheel to rotate forwards but not backwards. Use two nuts so that it can move freely but not come undone.

<Image leg LegAssembly.png>

The chassis front is the end without a hole. Fit the legs onto the servos so they face the front. Screw them in place using the G3 screws.

Cut a piece of the twin and earth cable to 65mm and remove the sleeving from two if the wires. Bend the copper wire to match the template. Fit the wire between the upper hole in the back of the legs and the chassis. Bend over the ends to fix in place. If you make the wire too long then it is possible to bend the wire slightly to adjust it. When the upper leg is straight down then the knee joint should be straight.

<image Wire Template.png>

Wire up the breakout board to the powerboost module. Ensure that the wires are long enough so that these can fit into the elephant’s body. Take care to check the polarization of the power connections. Put the switch on slightly longer wires so it will reach through the chassis.

Wire up the ESP32 board to the powerboost module. Fit the switch into the rear cutout in the chassis and hot glue into place.

Plug in the servo connectors.

Check your wiring before connecting up the battery.

DRIVING THE ELEPHANT

You'll need to configure your secrets file to your Wifi credentials, compile the code and upload it to the board. When the ESP32 boots it will connect to the Wifi and start a web server. Enable the servo power by sliding the switch. Your elephant should stand straight.

Point your web browser to http://esp32.local and you should see the control buttons. Pressing these should cause the elephant to move.

WHERE TO FROM HERE?

The design of the elephant is quite modular so you could swap out the head and tail to form other animals?
Perhaps wire up a 5th servo and have a moving trunk on your model?
Using the REST API of the elephant you could remote control the elephant programmatically?
If you don't fancy an animal then perhaps combine your 4 servos and control circuit into a different vehicle or a robot arm?