```
#include <curl.h>
#include "dist\json\json.h"
#include "dist\json\json-forwards.h"
#define CURL_STATICLIB
#include <cstdint>
#include <thread>
#include <iostream>
#include <memory>
#include <string>
#include <sstream>

Json::Value jsonData;
namespace
{
	std::size_t callback(const char* in, std::size_t size, std::size_t num, char* out)
	{
		std::string data(in, (std::size_t) size * num);
		*((std::stringstream*) out) << data;
		return size * num;
	}
}

int readJson()
{
	const std::string
url("http://192.168.0.105/kas/plcvariables?variables=ActualPos&format=json");

	CURL* curl = curl_easy_init();

	// Set remote URL.
	curl_easy_setopt(curl, CURLOPT_URL, url.c_str());

	// Don't bother trying IPv6, which would increase DNS resolution time.
	curl_easy_setopt(curl, CURLOPT_IPRESOLVE, CURL_IPRESOLVE_V4);

	// Don't wait forever, time out after 10 seconds.
	curl_easy_setopt(curl, CURLOPT_TIMEOUT, 10);

	// Follow HTTP redirects if necessary.
	curl_easy_setopt(curl, CURLOPT_FOLLOWLOCATION, 1L);

	// Response information.
	int httpCode(0);
	// std::unique_ptr<std::string> httpData(new std::string());

	std::stringstream httpData;

	// Hook up data handling function.
	curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, callback);

	// Hook up data container (will be passed as the last parameter to the
	// callback handling function).  Can be any pointer type, since it will
```

```
	// internally be passed as a void pointer.
	curl_easy_setopt(curl, CURLOPT_WRITEDATA, &httpData);

	// Run our HTTP GET command, capture the HTTP response code, and clean up.
	curl_easy_perform(curl);
	curl_easy_getinfo(curl, CURLINFO_RESPONSE_CODE, &httpCode);
	curl_easy_cleanup(curl);

	if (httpCode == 200)
	{
		std::cout << "\nGot successful response from " << url << std::endl;

		// Response looks good - done using Curl now.  Try to parse the results
		// and print them out.

		Json::CharReaderBuilder jsonReader;
		std::string errs;

		if (Json::parseFromStream(jsonReader, httpData, &jsonData, &errs))
			// jsonReader.parse(httpData, jsonData))
		{

			std::cout << jsonData["ActualPos"]["value"] << std::endl;


			std::cout << std::endl;
		}
		else
		{
			std::cout << "Could not parse HTTP data as JSON" << std::endl;
			std::cout << "HTTP data was:\n" << httpData.str() << std::endl;
			return 1;
		}
	}
	else
	{
		std::cout << "Couldn't GET from " << url << " - exiting" << std::endl;
		return 1;
	}

	return 0;
}
void sendJson() {

}
int main() {
//Read json data from PDMM controller
	readJson();
	return 0;
}
```