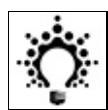


সূচিপত্র

পরিচিতি	0
প্রাথমিক ধারণা	1
আর্ডুইনো পরিচয়	1.1
আর্ডুইনো IDE ডাউনলোড, ইন্স্টল ও বোর্ড কনফিগারেশন	1.2
আর্ডুইনো বোর্ড ও কোড সম্পর্কে ধারণা	1.3
ডিজিটাল ইনপুট আউটপুট ও লুপ	1.4
পালস উইডথ মড্যুলেশন পরিচিতি	1.5
অবজেক্ট ওবিয়েটেড প্রোগ্রামিং পরিচিতি	1.6
অবজেক্ট ওবিয়েটেড প্রোগ্রামিং পরিচিতি (শেষ ভাগ)	1.7
সিরিয়াল কম্যুনিকেশন (সংক্ষেপ)	1.8
অ্যানালগ থেকে ডিজিটাল রূপান্তর (ADC)	1.9
ইন্টারফেস	2
আর্ডুইনোর সাথে LCD ইন্টারফেস (১ম অংশ)	2.1
লাইব্রেরি তৈরি করা ও ইনহেরিট্যান্স এর মাধ্যমে লাইব্রেরি মডিফাই করা	2.2
অ্যাডভান্সড	3
Qt দিয়ে Arduino এর জন্য ডেস্কটপ অ্যাপলিকেশন ডেভেলপমেন্ট	3.1
Python দিয়ে আর্ডুইনোর সাথে সিরিয়াল কমিউনিকেশন, আর্ডুইনো সিরিয়াল ইন্ডেক্স পরিচিতি ও PyQt4 দিয়ে ডেস্কটপ অ্যাপ ডেভেলপমেন্ট	3.2



howtocode.com.bd

পৃষ্ঠা লাইক করুন 10হাজার পছন্দগুলি

আপনার বয়সের মধ্যে আপনাই এটা প্রথম পছন্দ করুন



কোর্স এর মূল পাতা | HowToCode মূল সাইট | সবার জন্য প্রোগ্রামিং ব্লগ | পিডিএফ ডাউনলোড

বাংলায় Arduino ডিতিক এন্ডেড সিস্টেম এর খুঁটি নাটি

[gitter](#) [join chat](#)



manashmndl 80



howtocode-com-bd 4



nuhil 2



gitter-badger 1

সংক্ষেপ

কিছুদিন আগেও মাইক্রোকন্ট্রোলার হাতে নিয়ে সবকিছু কন্ট্রোল করা খুব একটা সহজ কাজ ছিল না।

Microcontroller হল এমন একটি Programmable Platform যেটা দিয়ে আপনি জটিল সব মেকানিক্যাল, ইলেক্ট্রিক্যাল কিংবা সফটওয়্যার সিস্টেম কন্ট্রোল করতে পারবেন কিছু কমান্ডের মাধ্যমে। মাইক্রোকন্ট্রোলার দিয়ে অসংখ্য কাজ করা যায়। ডিজিটাল ইলেক্ট্রনিক্স, প্রোগ্রামিং কিংবা ইউম্যান কম্পিউটার ইন্ট্যারাকশনের জন্য আডুইনো এখন বেস্ট চয়েস। যদি আপনার এমবেডেড সিস্টেম সম্পর্কে একদমই ধারণা না থাকে তাহলে আডুইনো দিয়ে শুরু করা ভাল। যেকোন প্রজেক্ট, রোবটিক্স, ইটারনেট অফ থিংস ইত্যাদি নানা কাজে আডুইনো বোর্ড ব্যবহাত হচ্ছে। তার মূল কারণ হল এর সমন্বয় লাইব্রেরি ও সহজবোধ্যতা। প্রোগ্রামিং সম্পর্কে একটু আধুনিক ধারণা আর বেসিক ইলেক্ট্রনিক্স জ্ঞান থাকলেই আডুইনো বোর্ড ব্যবহার করে সহজেই চমৎকার সব প্রজেক্ট তৈরি করা সম্ভব।

কোস্টি কাদের জন্য?

আডুইনো নিয়ে কাজে উৎসাহী সবার জন্যই মূলত সিরিজটি। যারা শুরু করছেন তাঁদের শেখার কাজে ও যারা অ্যাডভান্সড লেভেলে আছেন তাদের বেফারেজের জন্য লাগতে পারে। এই সিরিজে ধারণা করা হবে আপনার বেসিক প্রোগ্রামিং ও বেসিক ইলেক্ট্রনিক্স জ্ঞান আছে। এখানে কিছু বিষয় নিয়ে অতিরিক্ত আলোচনা করা হতে পারে যেগুলো যদিও আডুইনো কীভাবে কাজ করে তার সাথে পুরোপুরি সম্পৃক্ত থাকবে না কিন্তু টেকনিক্যাল টপিকগুলোর উপরে পরিষ্কার ধারণার জন্য আলোচ্য বিষয় কাজে লাগতে পারে।

কী কী আলোচনা করা হবে এই কোর্সে?

এই সিরিজটা রেসিপি বইয়ের মত হবে না। যদি আপনি স্টেপ বাই স্টেপ ইন্ট্রাকশন চান যে একটি প্রয়েষ্ঠ পুরোপুরি তৈরি করতে কী কী লাগবে এবং কীভাবে করতে হবে, তাহলে সিরিজটি আপনার জন্য নয়। সিম্যুলেশন ও প্রাথমিক জিনিসগুলো স্টেপ বাই স্টেপ অবশ্যই দেখানো হবে তাই বলে একটি পূর্ণাঙ্গ প্রজেক্টের সবচুকু কখনোই দেওয়া হবে না। এই সিরিজের মাধ্যমে বেসিক ইলেক্ট্রিক্যাল, কম্পিউটার সায়েন্স, প্রোগ্রামিং এবং উচ্চ পর্যায়ের চিত্তার খোরাক যোগানো হবে যেটা আডুইনোর মাধ্যমে আপনি আয়ত্তে আনতে পারবেন।

যখন সাধারণ কোন আডুইনো প্রজেক্ট নিয়ে আলোচনা করা হবে, তখন শুধু কম্পোনেন্টগুলো দিয়ে কীভাবে সার্কিট তৈরি করে তা-ই শিখবেন না বরং কীভাবে Schematic design পড়তে হয়, কেন ও যন্ত্রপাতিগুলো ব্যবহাত হল সেটা জানতে পারবেন তার পশাপাশি Datasheet কী কাজে লাগে ও কীভাবে আপনি Datasheet ব্যবহার করে নিজের প্রজেক্ট অনুযায়ী কম্পোনেন্ট ব্যবহার করবেন সেটাও জানতে পারবেন। আমি সিম্পল কোডগুলো ব্যাখ্যাসহ লিখে দিব, কিন্তু মূল প্রজেক্টের কোডিং আপনাকেই করতে হবে।

- আডুইনো বেসিক
 - আডুইনো পরিচয়
 - আডুইনো IDE ডাউনলোড, ইন্ষটল ও বোর্ড কনফিগারেশন
 - আডুইনো বোর্ড ও কোড সম্পর্কে ধারণা
 - ডিজিটাল ইনপুট আউটপুট ও লুপ
 - পালস উইডথ মড্যুলেশন পরিচিতি
 - অ্যানালগ থেকে ডিজিটাল রূপান্তর (ADC)
 - অবজেক্ট ওরিয়েন্টেড প্রোগ্রামিং পরিচিতি
 - সিরিয়াল কম্যুনিকেশন
- ইন্টারফেস
 - আডুইনোর সাথে LCD ইন্টারফেস
 - আডুইনোর সাথে Keypad ইন্টারফেস
 - আডুইনো ও L293D দিয়ে মোটর ড্রাইভিং
 - আডুইনো ও Relay
 - LCD, Keypad, LED, Motor এর কঞ্জিনেশনে জটিল প্রজেক্ট তৈরি
- অ্যাডভাঞ্চড
 - Java, C#, QtC++, Processing এর মাধ্যমে সিরিয়াল কম্যুনিকেশন
 - প্যাটার্ন অ্যালগরিদম দিয়ে Line Follower Robot তৈরি
 - Wall Follower/Avoider Robot তৈরি
 - PID অ্যালগরিদম দ্বারা LFR তৈরি
 - Bluetooth Controlled Robot তৈরি
 - Wifi Controlled Robot তৈরি

ওপেন সোর্স

এই বইটি মূলত শ্বেচ্ছাপ্রমে লেখা এবং বইটি সম্পূর্ণ ওপেন সোস। এখানে তাই আপনিও অবদান রাখতে পারেন লেখক হিসেবে। আপনার কন্ট্রিভিউশান গৃহীত হলে অবদানকারীদের তালিকায় আপনার নাম যোগ করে দেওয়া হবে।

এটি মূলত একটি [গিটহাব রিপোজিটোরি](#) যেখানে এই বইয়ের আর্টিকেল গুলো মার্কডাউন ফরম্যাটে লেখা হচ্ছে। রিপোজিটরিটি ফর্ক করে পুল রিকুয়েস্ট পাঠানোর মাধ্যমে আপনারাও অবদান রাখতে পারেন।

 Like 46  Share



This work is licensed under a [Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License](#).

প্রাথমিক ধারণা

আজকে তেমন কিছু আলোচনা করা হবে না। কোস্টি কম্পিউট করার জন্য আপনার যেসব জিনিস দরকার হবে সেগুলোর কথা আর আডুইনোর ব্যাপারে জানব। তাহলে শুরু করা যাক

যেসব যন্ত্রপাতি লাগবে:

সিম্যুলেশনের জন্য:

যদের আপাতত কিছু কেনা সম্ভব হচ্ছে না ইচ্ছা করলে তারাও আডুইনো শিখতে পারেন, এটা শেখার জন্য আডুইনো বোর্ডের প্রয়োজন নেই। Proteus 8.X SPY Design Suite [X & Y for version number]

রিয়েল লাইফ প্র্যাটিস:

প্র্যাটিক্যাল ও ভার্চুয়াল সিম্যুলেশনের মধ্যে তফাত থাকা অস্বাভাবিক নয়। এম্বেডেড সিস্টেমে যতটা পারা যায় ততটা প্র্যাটিক্যাল ওরিয়েন্টেড হওয়াই ভাল। প্রতিটি পোর্টে বলে দেওয়া হবে কী কী যন্ত্রপাতি লাগবে। তারপরও যেসব যন্ত্রপাতি প্রায় প্রয়েক্টেই দরকার হবে সেগুলো বলা হল:

- Arduino UNO R3 or Arduino Mega 2560
- Half Size Breadboard
- Resistor
- Capacitor
- LED
- Variable Resistor
- DMM (Digital Multimeter)

আডুইনো কী?

Arduino এর সবচেয়ে বেস্ট পার্ট হল আপনি যেটা চান সেটাই আডুইনো। কথাটা একটু গোলমেলে লাগলেও আরেকটু খোলাখুলি বলা যাক। যদি চান তাহলে আপনার আডুইনো প্ল্যাটফর্মটি হতে পারে আপনার চারাগাছে পানি সরবরাহ করার সিস্টেম, হতে পারে ওয়েব সার্ভার কিংবা কোয়াডকপ্টার অটোপাইলটও হতে পারে।

Arduino একটি মাইক্রোকন্ট্রোলার ডেভেলপমেন্ট প্ল্যাটফর্ম যেটা Intuitive Programming Language এ যুক্ত এবং আপনি এর IDE দ্বারা ডেভেলপ করতে পারেন। আডুইনোর সাথে Sensor, actuator, light, speaker, add-on module (যেটা Arduino Shield নামেও পরিচিত) এবং অন্যান্য IC ব্যবহার করে যেকোন সিস্টেমের Programmable Brain হিসেবে সেট করতে পারেন আপনার আডুইনোকে।

Arduino এর দ্বারা যা কিছু সম্ভব তা একটি কোর্সে লেখা অসম্ভব। ক্রিয়েটিভিটির লিমিট হয় না, যেহেতু আডুইনো ক্রিয়েটিভিটি প্রকাশ করারই একটা মাধ্যম তাই এর দ্বারা কতকিছু সম্ভব সেটারও সীমা নাই। এই কোস্টি কেবল সাধারণ ধারণা দিবে এবং বেসিক স্কিল আয়ত্ত করার পথ দেখাতে পারে, এর বেশি কিছু আশা করা ভুল হবে।

আডুইনো : একটি ওপেন সোর্স প্ল্যাটফর্ম

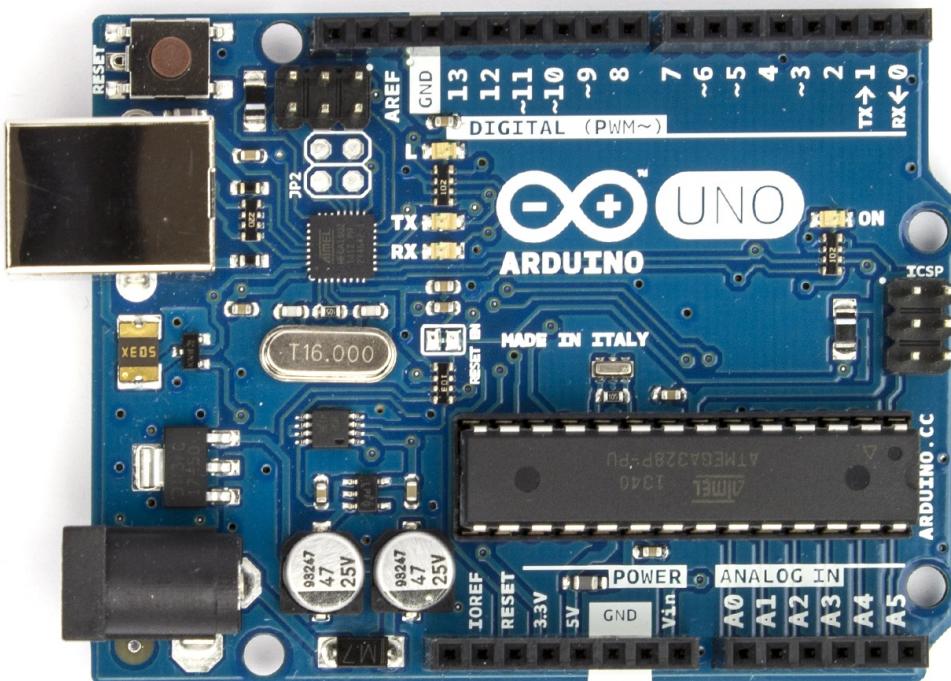
আপনি যদি ওপেন সোর্স নবাগত হয়ে থাকেন তাহলে আডুইনোর মাধ্যমেই বুকবেন ওপেন সোর্স সবকিছু এতটা পাওয়ারহুল কেন! ওপেন সোর্স নিয়ে নতুন করে বকবকানির কিছু নেই। যেহেতু আডুইনো একটি ওপেন সোর্স হার্ডওয়্যার তাই এর ডিজাইন ফাইল, স্কিম্যাটিক্স ও সোর্স কোড সবই উন্মুক্ত। এর মানে শুধু যে আপনি আডুইনো হ্যাক করে বিভিন্ন কাজ করবেন তাই নয়, বরং আপনি যদি ডিজিটাল ইলেক্ট্রনিক্সের গুরু হয়ে থাকেন তাহলে খুব সহজই আপনার মনমত জিনিসপত্র অ্যাড করে নিজস্ব প্ল্যাটফর্ম তৈরি করতে পারবেন।

আডুইনোর ডিজাইন, সোর্স কোড, লাইব্রেরি ডেক্সিপশন ও লাইব্রেরি ডাউনলোড করা যাবে তাদের অফিশিয়াল সাইট [আডুইনো.সিসি](http://arduino.cc) থেকে।

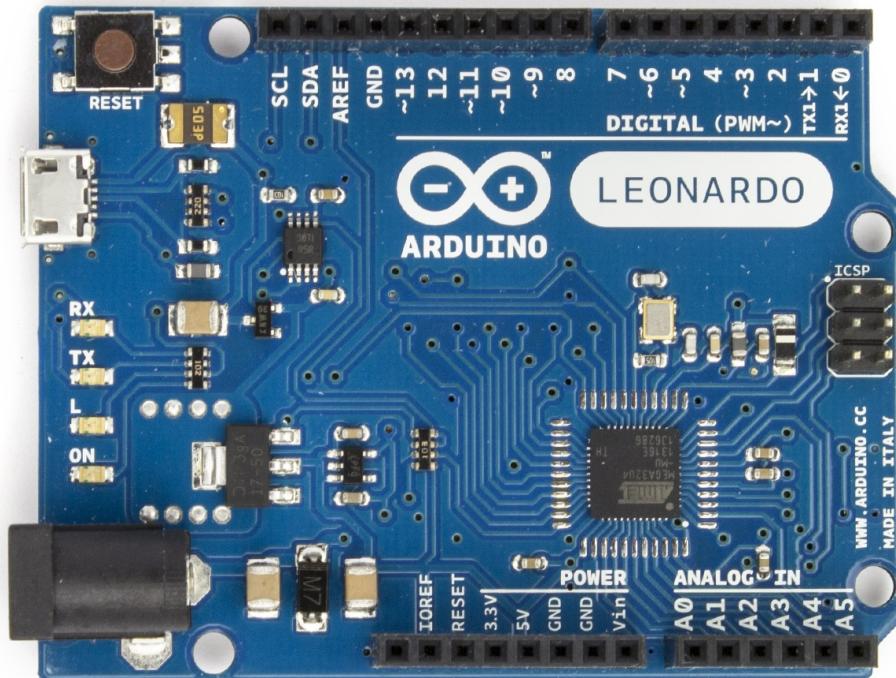
আডুইনো অরিজিনাল বোর্ডের দাম প্রায় ২০০০ – ৩০০০ এর মত। সেখানে Chinese ক্ষেত্রে ডার্সন পাওয়া যায় মাত্র ৯০০ (UNO)-১২০০ (MEGA) টাকার মধ্যে। আডুইনো ওপেন সোর্স বলেই এটা সম্ভব। Arduino ছাড়াও এইরকম ক্ষেত্রে প্ল্যাটফর্ম আছে আরও ফাংশনালিটি সহ যেমন Freeduino, pcDuino, ODOO ইত্যাদি।

বিভিন্ন ধরণের আডুইনো বোর্ড:

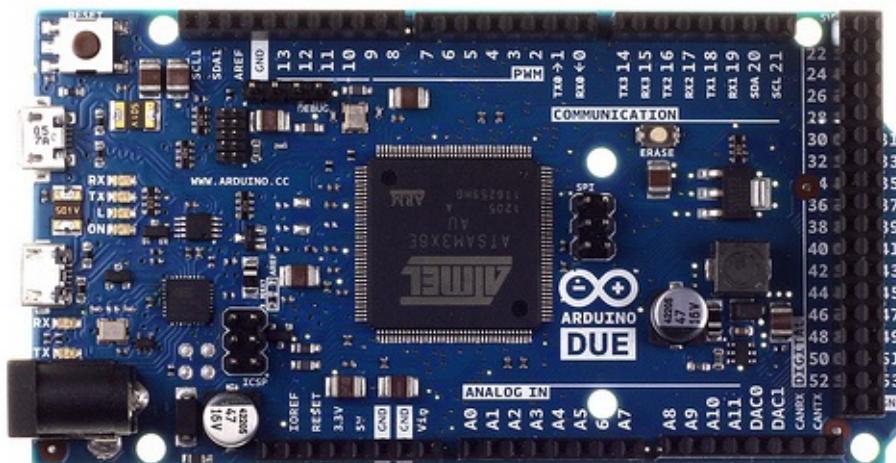
Arduino UNO R3 (R stands for Revision):



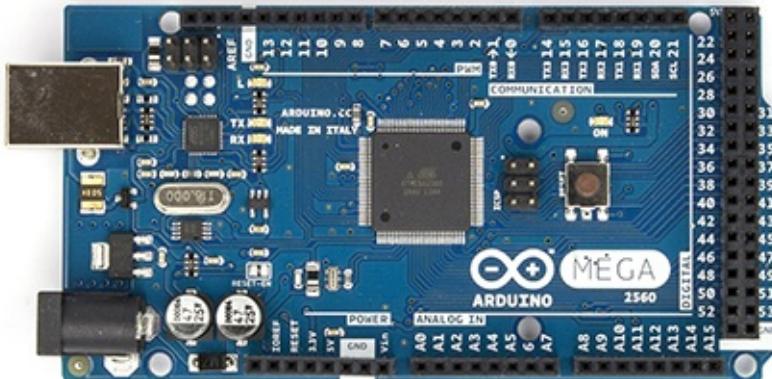
Arduino Leonardo:



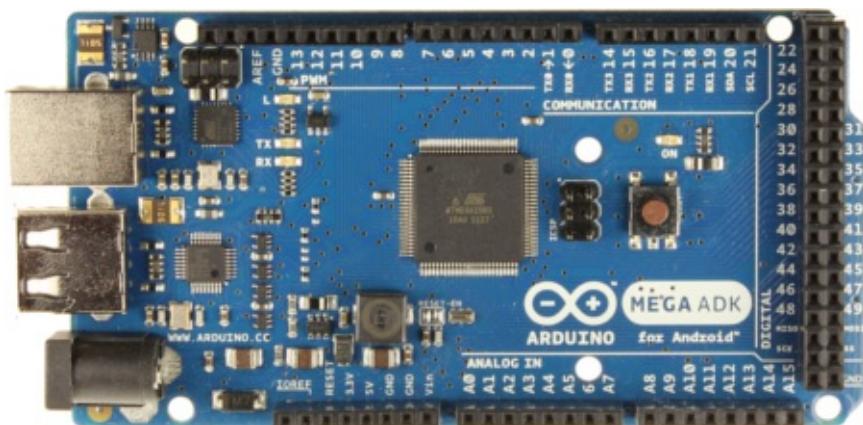
Arduino DUE:



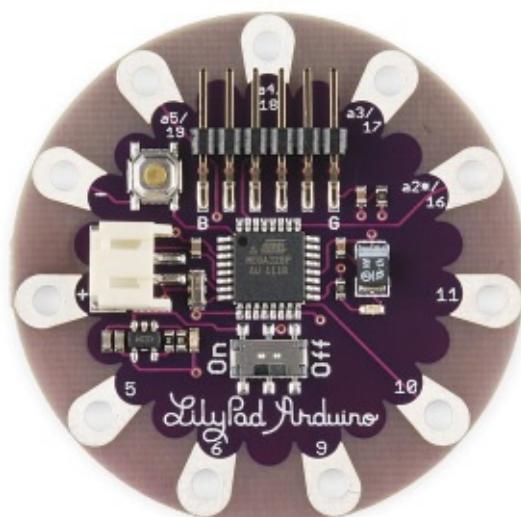
Arduino Mega 2560:



Arduino Mega ADK [Android Development Kit]:



Arduino LilyPad:



Arduino Yun:



পরবর্তী পর্বে আমরা আডুইনো IDE ডাউনলোড ও কনফিগার করব।

আর্ডুইনো IDE ডাউনলোড, ইন্টল ও বোর্ড কনফিগারেশন এবং কিছু সাধারণ সমস্যার সমাধান

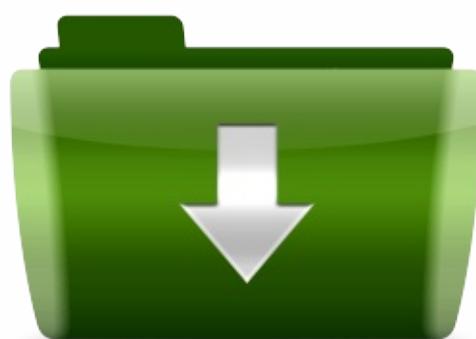
এই পোস্টের জন্য আপনার যা দরকার হবে [যদি আর্ডুইনো বোর্ড সংগ্রহ করতে ব্যর্থ
হন তাহলে সিম্যুলেশন বেজড পোস্টের জন্য পরবর্তী পোস্ট থেকে শুরু করুন]

- প্রায় 100MB ডেটাসহ ইন্টারনেট কানেকশন [যদি ডাউনলোড না করতে চান সেক্ষেত্রে আপনাকে IDE যোগাড়
করে নিতে হবে]
- Arduino UNO R3 অথবা Arduino Mega 2560 [এখানে আমি Arduino Mega 2560 ব্যবহার করব,
ভয়ের কিছু নেই দুইটার কনফিগার করার পদ্ধতি একই রকম]
- USB A – B Cable [আপনার ক্রয়কৃত আর্ডুইনোর সঙ্গে যে কেবলটি দেওয়া আছে সেটা]

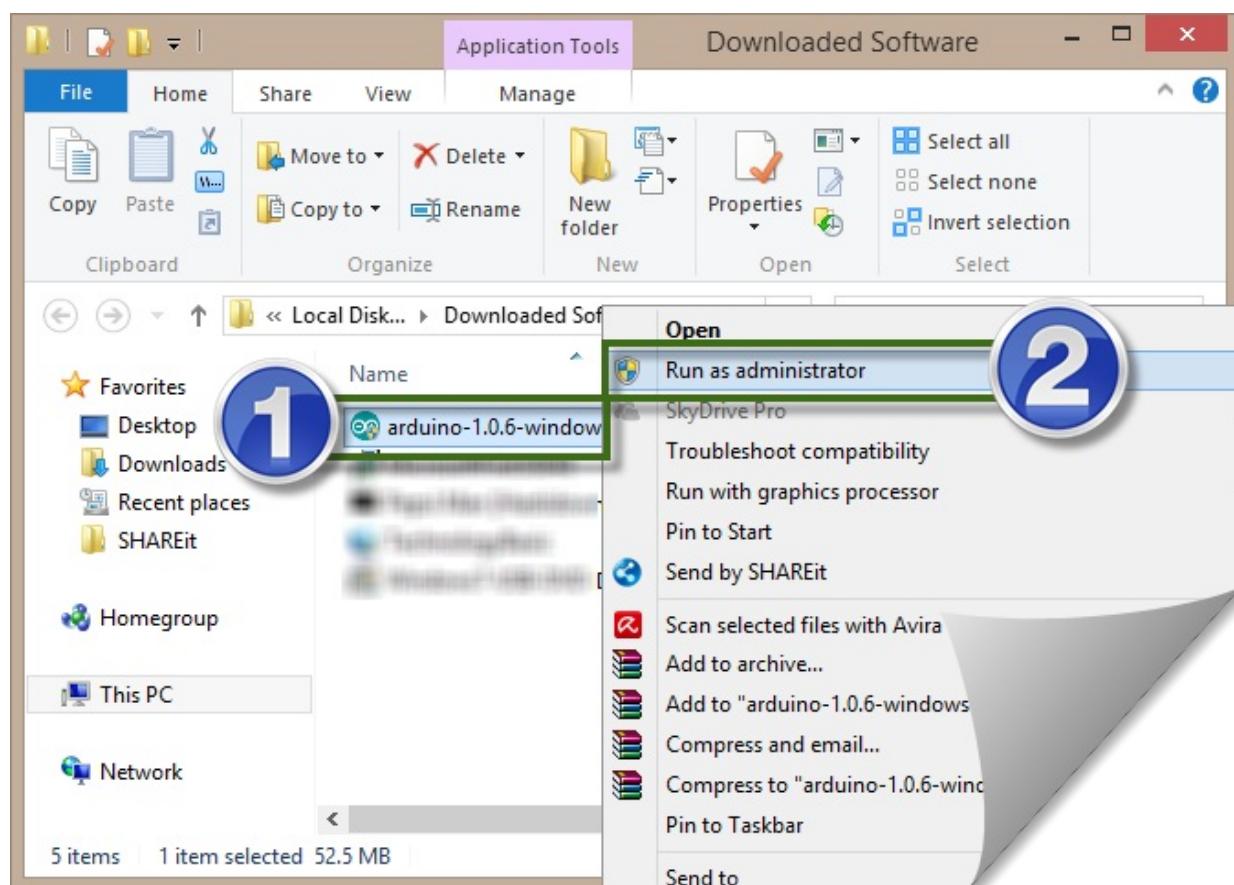
দ্রষ্টব্য: টিউটোরিয়ালটি উইন্ডোজ অপারেটিং সিস্টেম ইউজারদের জন্য। লিনাক্সে কোন ঝামেলা করা লাগে না,
লিনাক্সের জন্য এখান থেকে ([32 bit](#), [64 bit](#)) IDE ডাউনলোড করুন আর আর্ডুইনো পিসির সাথে কানেক্ট
করুন। সমস্য এড়ানোর জন্য টার্মিনাল থেকে gksudo arduino কমান্ড দ্বারা আর্ডুইনো চালু করুন।

Arduino IDE ডাউনলোড ও ইন্টলেশন:

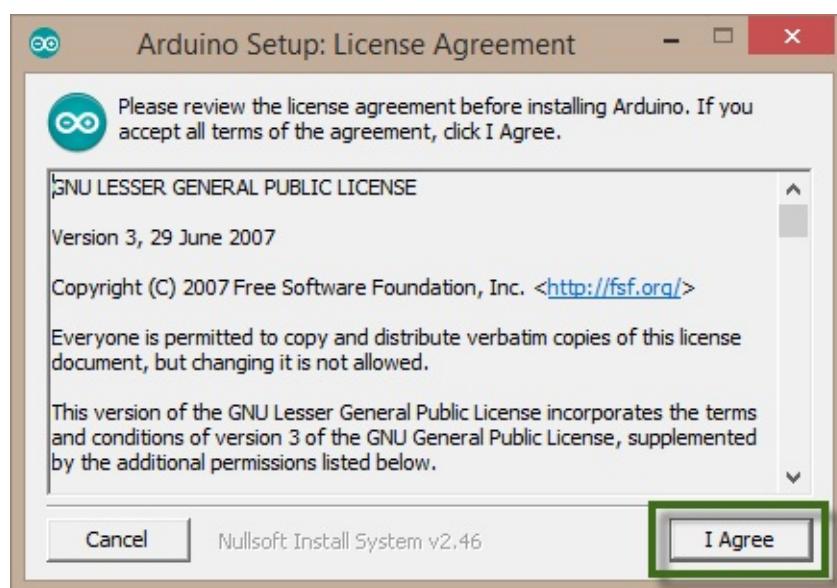
আর্ডুইনোর জন্য কোড লেখা, আপলোড করা ও সিরিয়াল মনিটরিংয়ের জন্য আপনার অফিশিয়াল IDE ডাউনলোড
করতে হবে। ওপেনসোর্স হওয়ার কারণে IDE টা জমকালো না তবে কাজ চলে আরকি। যদি IDE এর ইন্টারফেস
আপনার পছন্দ না হয় সেক্ষেত্রে Sublime Text Editor কে কীভাবে আর্ডুইনো IDE হিসেবে ব্যবহার করা যায় সেটা
দেখানো হবে। তাহলে কাজ শুরু করা যাক। ছবিতে ক্লিক করে ডাউনলোড করুন:



- ডাউনলোডেড ফাইলটা Run As Administrator হিসেবে চালু করুন:



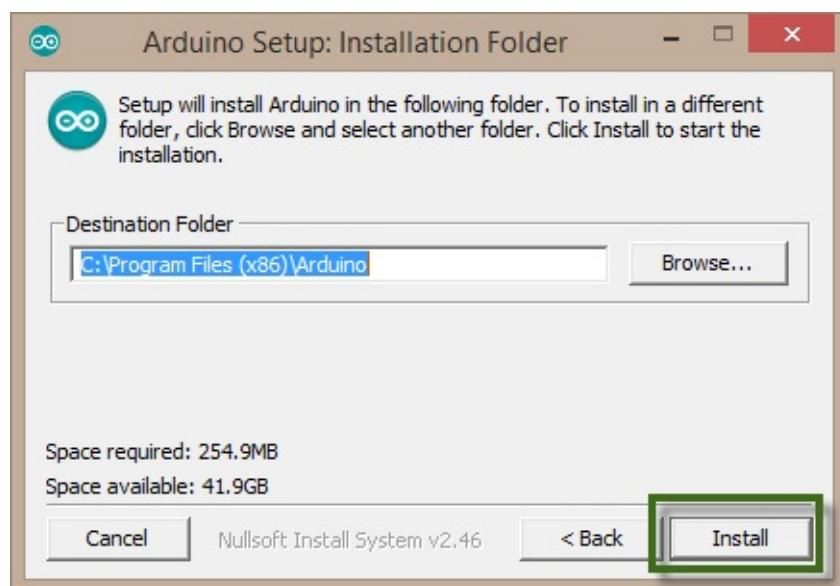
- I Agree:



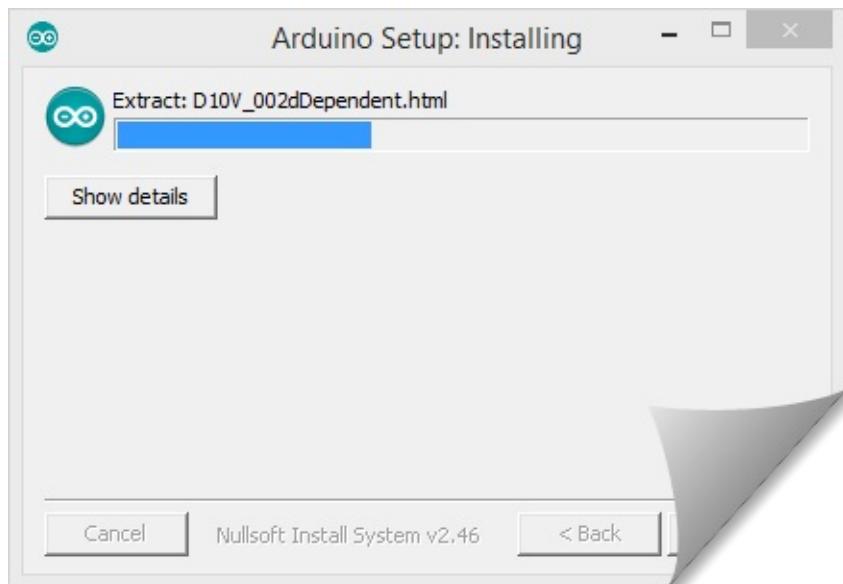
- Next:



- **Install:**



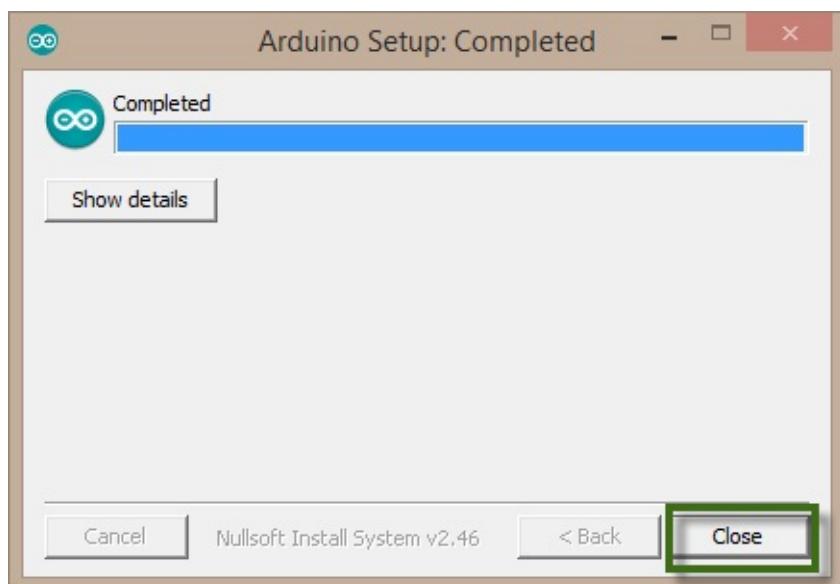
- **অপেক্ষা করন:**



- **Install এ ক্ষিকান:**

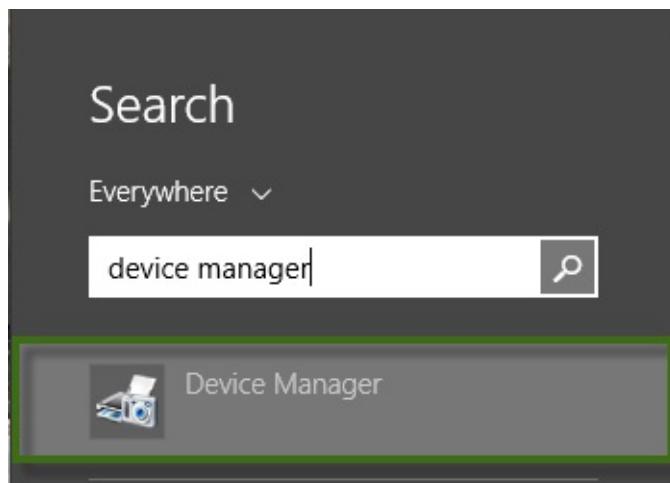


- **Close চাপুন:**



Arduino Board ড্রাইভার সেটাপ:

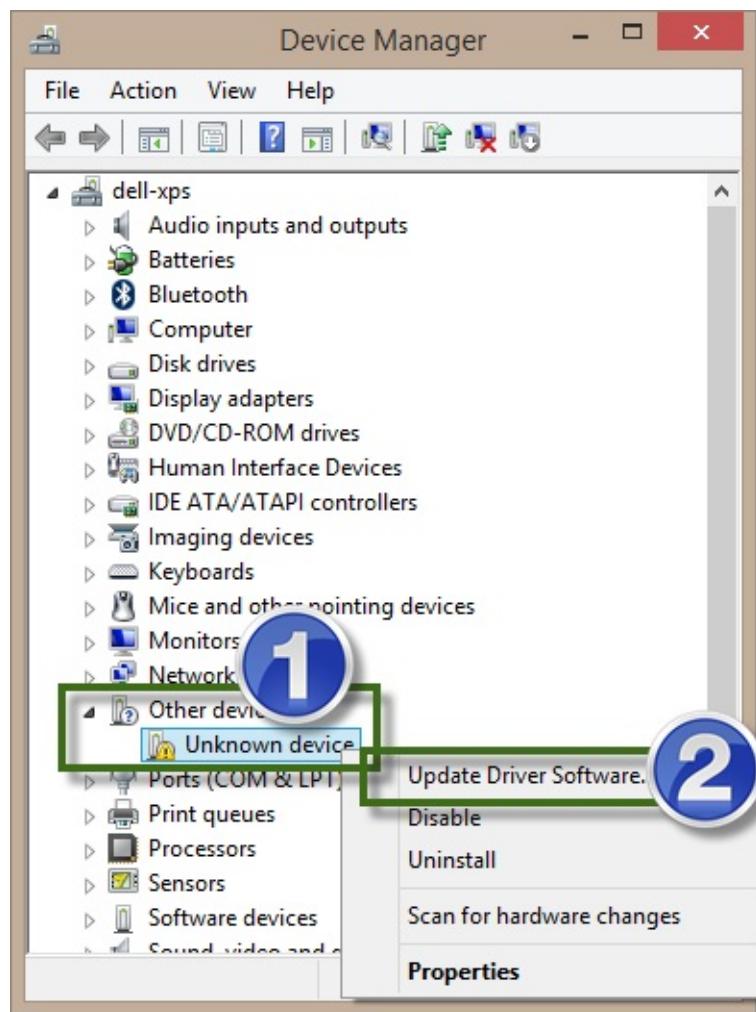
- আপনার আর্ডুইনো বোর্ডটি পিসির সাথে কানেক্ট করে উইন্ডোজের Device Manager ওপেন করুন:



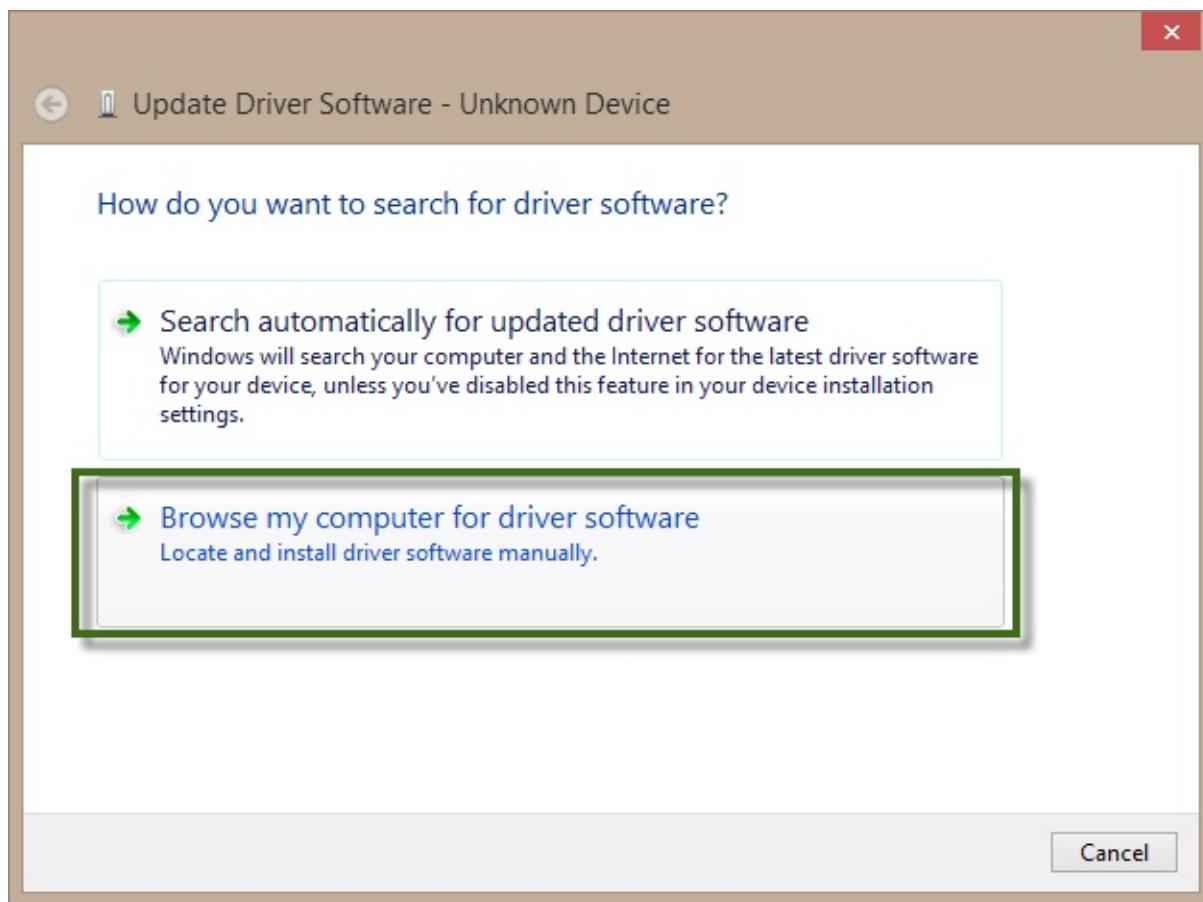
- এবার Ports এ ক্লিক করলে যদি Arduino দেখায় তাহলে বুঝবেন ড্রাইভার অটো ইন্�স্টল হয়ে গেছে:



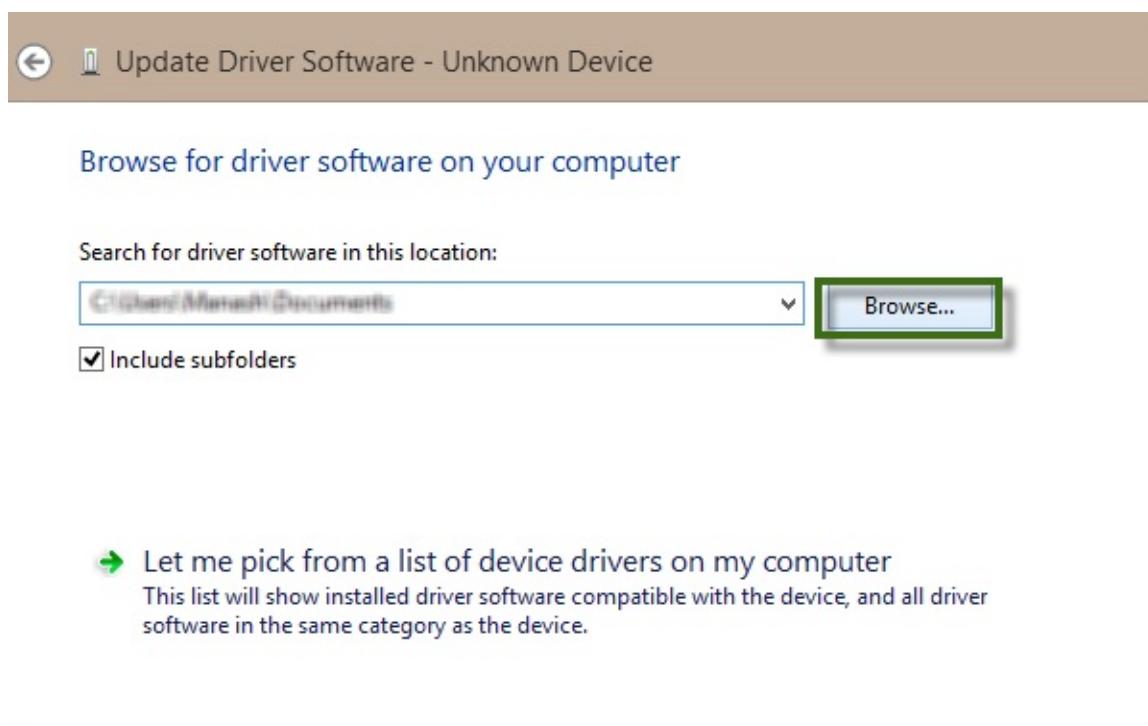
- আর যদি তা না দেখা যায় তাহলে Other devices এ যতগুলো Unknown Device আছে সেগুলোতে নিচের পদ্ধতিতে আর্ডুইনো ড্রাইভার সেটাপ দেওয়া শুরু করবেন। একটা না একটাতে সেটাপ হবেই :P
- Unknown Device এ রাইট ক্লিক করে Update Driver Software দিন:



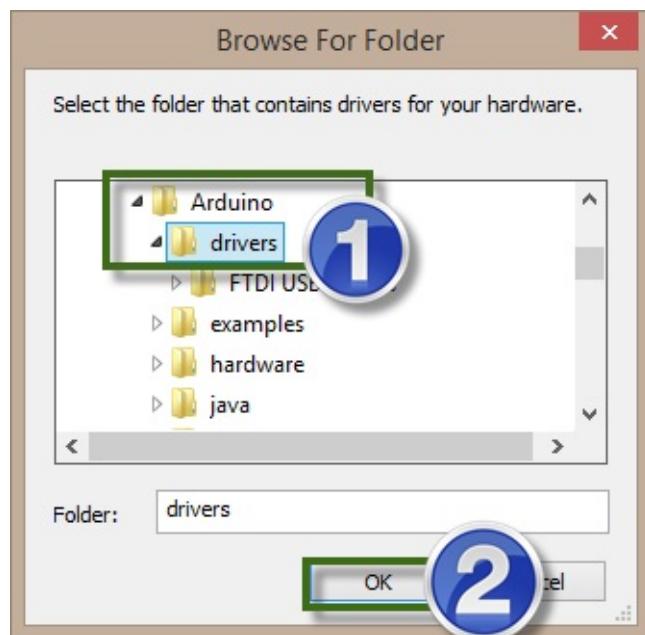
- Browse my computer for driver software এ ছিক করুন:



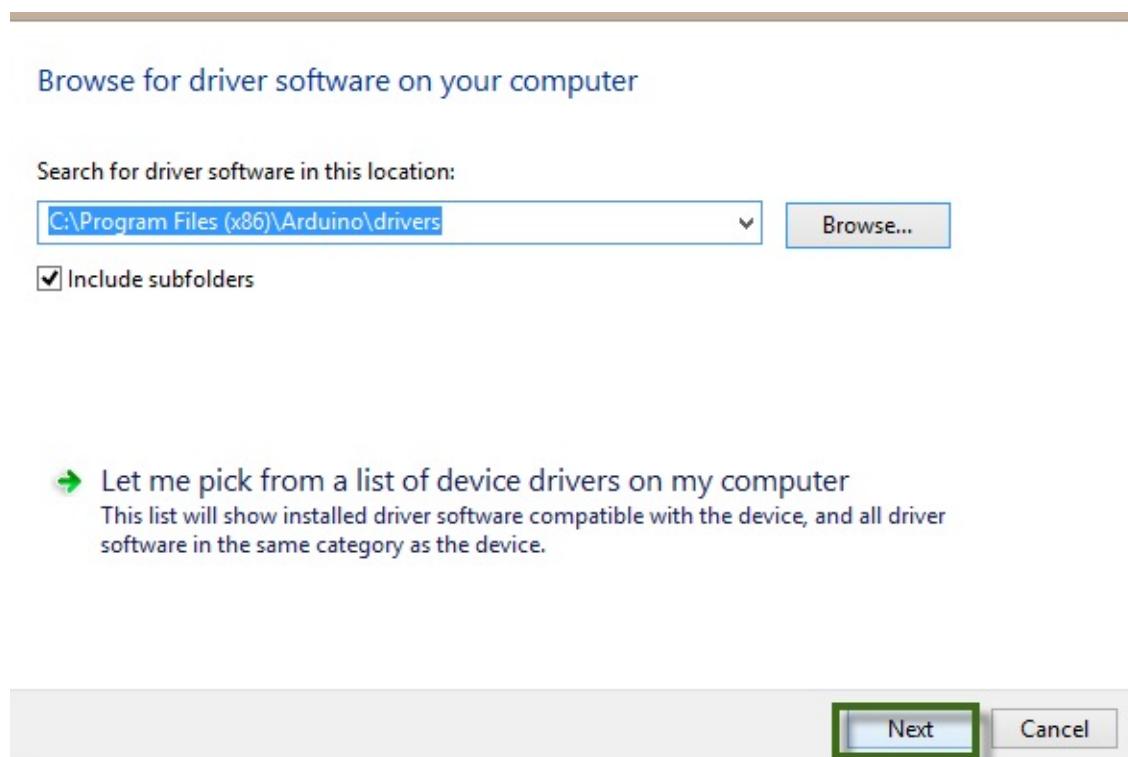
- Browse এ ক্লিকান:



- Arduino Folder টি খুঁজে বের করে Driver ফোল্ডারটি সিলেক্ট করে OK দিন:



- Next:

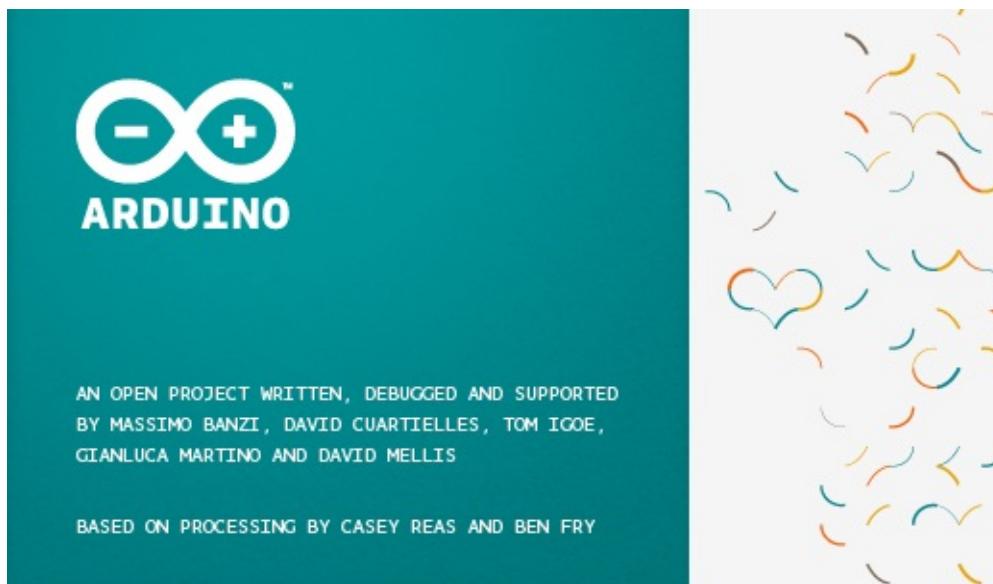


- এবার ওকে ওকে নেক্সট যা আসবে সবগুলোই দিন, যদি বলে যে This driver software is incompatible for the device বা এই জাতীয় কথা তাহলে অন্য Unknown Device এ একই পদ্ধতিতে Driver আপডেট করার ট্রাই করুন।

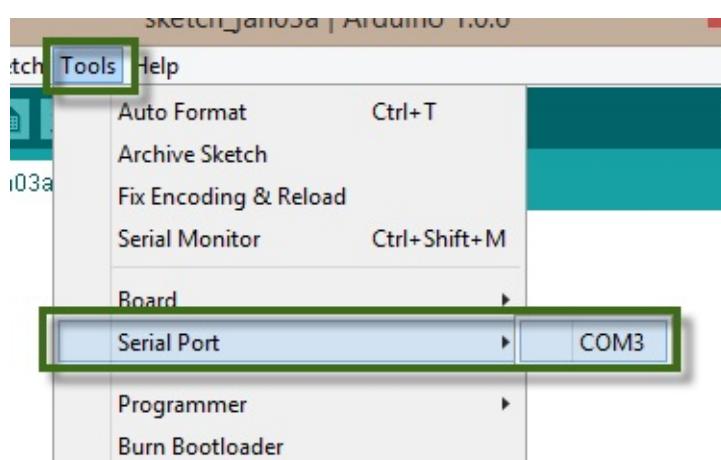
Arduino Configure করা ও প্রথমবারের মত Program বান্ন করা:

- বোর্ডের ড্রাইভার ঠিকঠাক সেটাপ হয়ে গেছে? তাহলে দেরি না করে প্রোগ্রাম আপলোড করে ফেলুন ষটপট। প্রোগ্রাম আপ্লোড করার আগে আর্ডুইনো বোর্ড ও COM Port সিলেক্ট করে নিতে হবে।

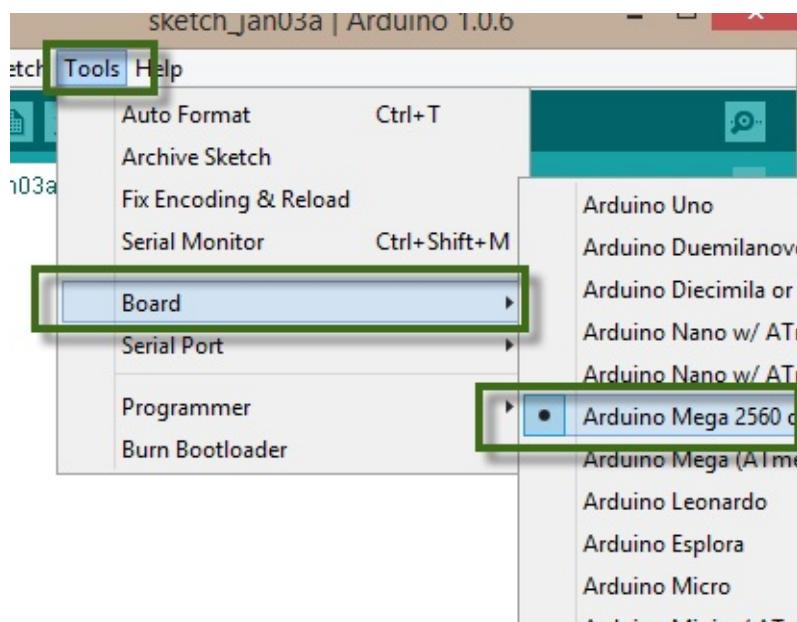
- Arduino IDE ওপেন করুন, নিচের শপ্ল্যাশ দেখাবে:



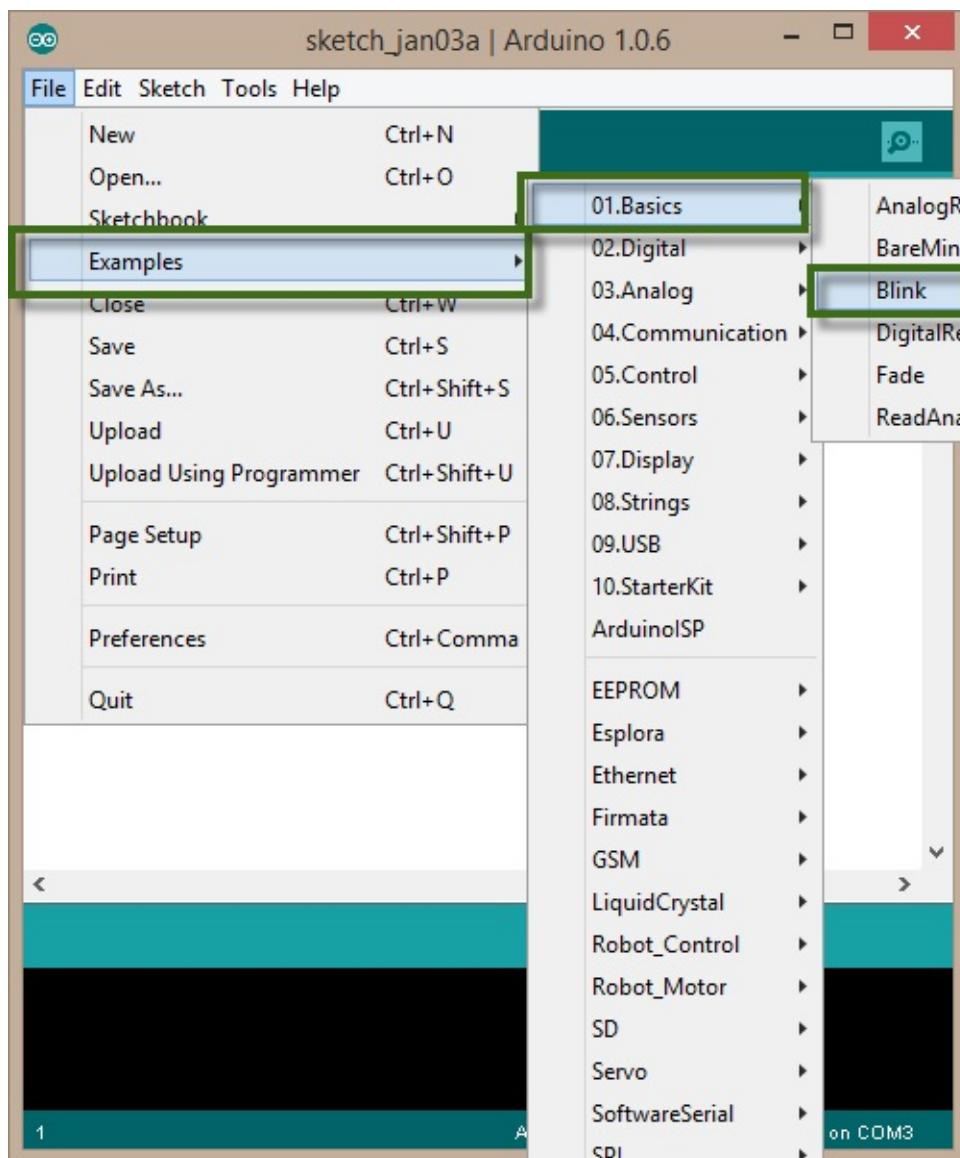
- এবার Tools > Serial Port > COMX [Device Manager এর Port থেকে দেখে নিতে পারেন আপনার আডুইনো কোন Port এ আছে]



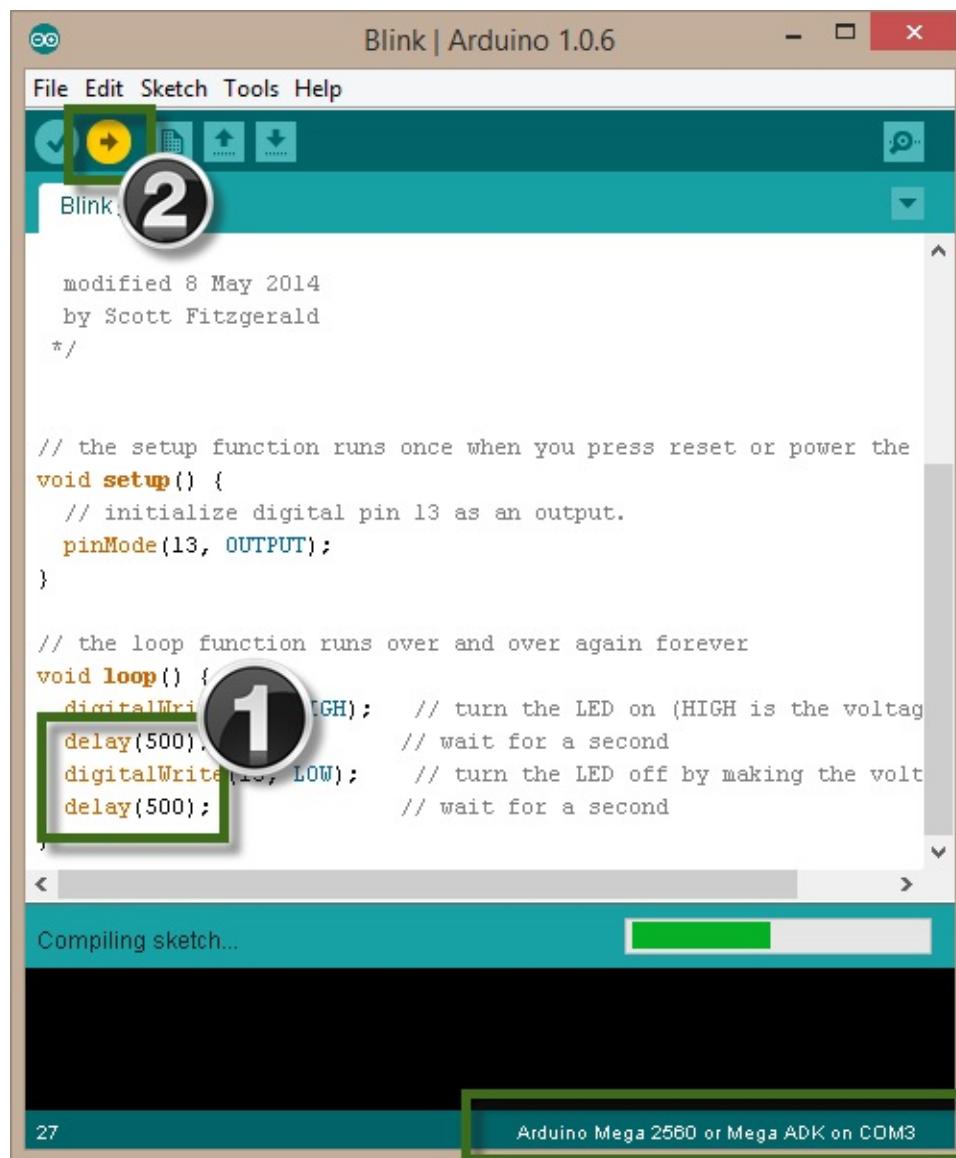
- আবার Tools > Board > Arduino [এখানে আপনার বোর্ডটি সিলেক্ট করবেন, আপনার বোর্ড UNO হলে সবার প্রথমটা আর Mega হলে ছবিতে যেটা দেখানো হয়েছে সেটা সিলেক্ট করলেই হবে]



- এখনো তো আডুইনো কোডিং শেখা হয় নি? সমস্যা নাই, IDE তেই শখানক Example দেওয়া আছে, তার যেকোনটা আপ্লোড দিলেই চলে। কিন্তু আপনার Arduino যে কোড অনুযায়ী কাজ করছে সেটা দেখাব জন্য Blink নামের Example টি বেশ কার্যকর। Arduino UNO, Mega তে 13 নাষ্ঠার পিনের সাথে একটি LED যুক্ত থাকে। তাই আমরা External Led ছাড়াও Led Blinking এর কোড কাজ করছে কিনা সেটা দেখতে পারি।
- Example এর জন্য File > Examples > Basics > Blink এ ক্লিক করুন, নতুন উইンドো ওপেন হবে:



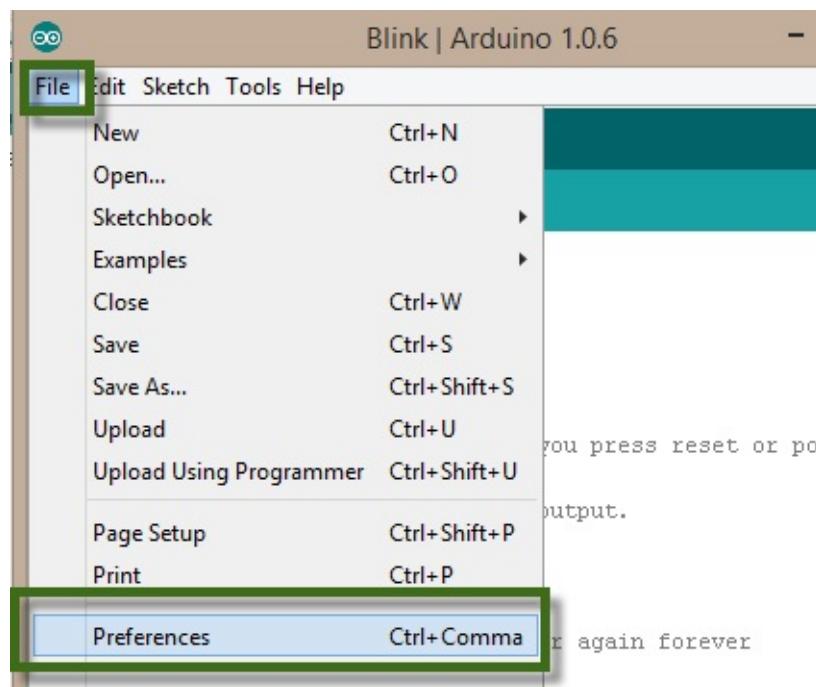
- Arduino তে আগে থেকেই Blink প্রোগ্রামটি বান্ন করা থাকে, তাই আমার পরামর্শ হল Blink প্রোগ্রামটি বান্ন করার আগে delay এর মধ্যে 1000 এর বদলে আপনি অন্য কোন সংখ্যা বসান, যেমন delay(100) বা delay(50) যাতে আমরা নতুন প্রোগ্রাম আর পুরনোটার মধ্যে তফাত খুঁজে পাই।
- [নোট: delay(int) ফাংশনটির আর্গুমেন্ট ইন্টিজার টাইপ এবং এখানে যেটা দেওয়া হবে তার একক মিলিসেকেন্ড, অর্থাৎ 1000 এর মানে হল 1 second]
- এরপর Upload বাটনটিতে ক্লিক করুন:



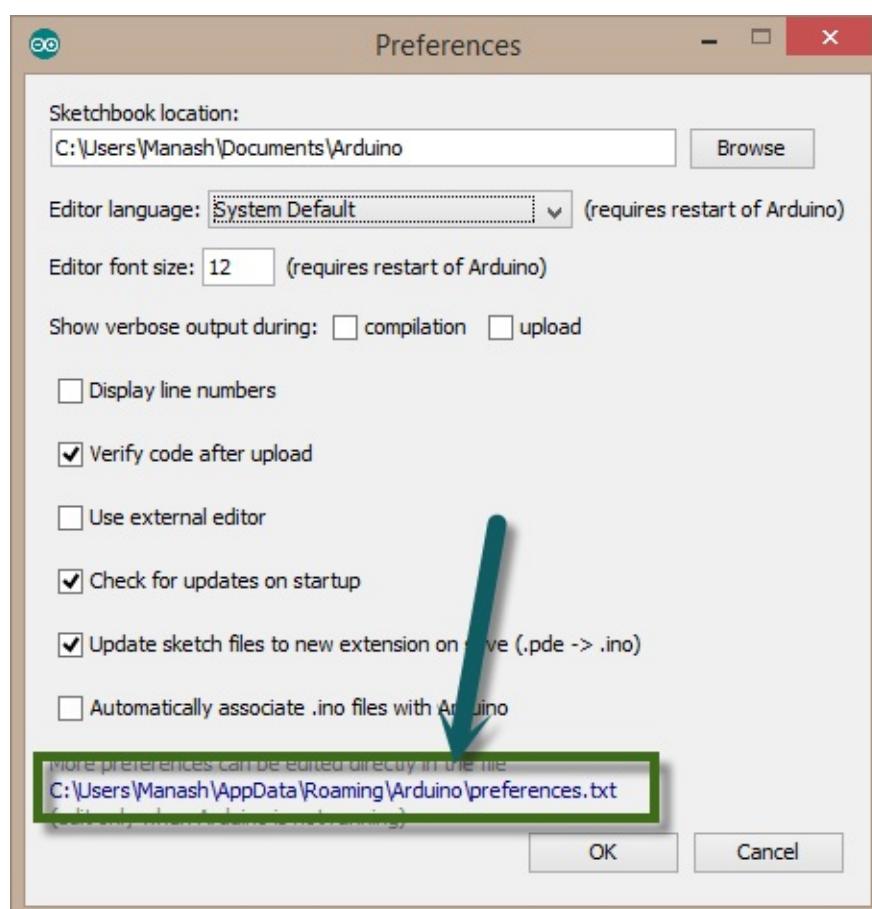
- এবার যদি লিংকিং দেখতে পান বা লিংকিংয়ে ভিন্নতা নজরে আসে অথবা যদি Arduino IDE তে দেখেন কোন ঝামেলা ছাড়াই Upload Completed আসে তাহলে বুঝতে হবে কোড আপলোড হয়েছে!

Arduino IDE কাস্টোমাইজেশন:

- আডুইনোর সবই ভাল খালি IDE টাই পছন্দ হ্য না। ফন্টের অবস্থা ভাল না, মাউজ স্ক্রল করে জুম করা যায় না আরও অনেক সমস্যা। যাই হোক, কাস্টোমাইজেশন বলতে এখানে ফন্ট কিভাবে পরিবর্তন করবেন সেটাই দেখানো হয়েছে, Heavy Customization এর জন্য গুগলিং করাই ভাল হবে।
- ফন্ট চেঞ্জ করার জন্য অনেক কাজ করতে হবে, File > Preferences:

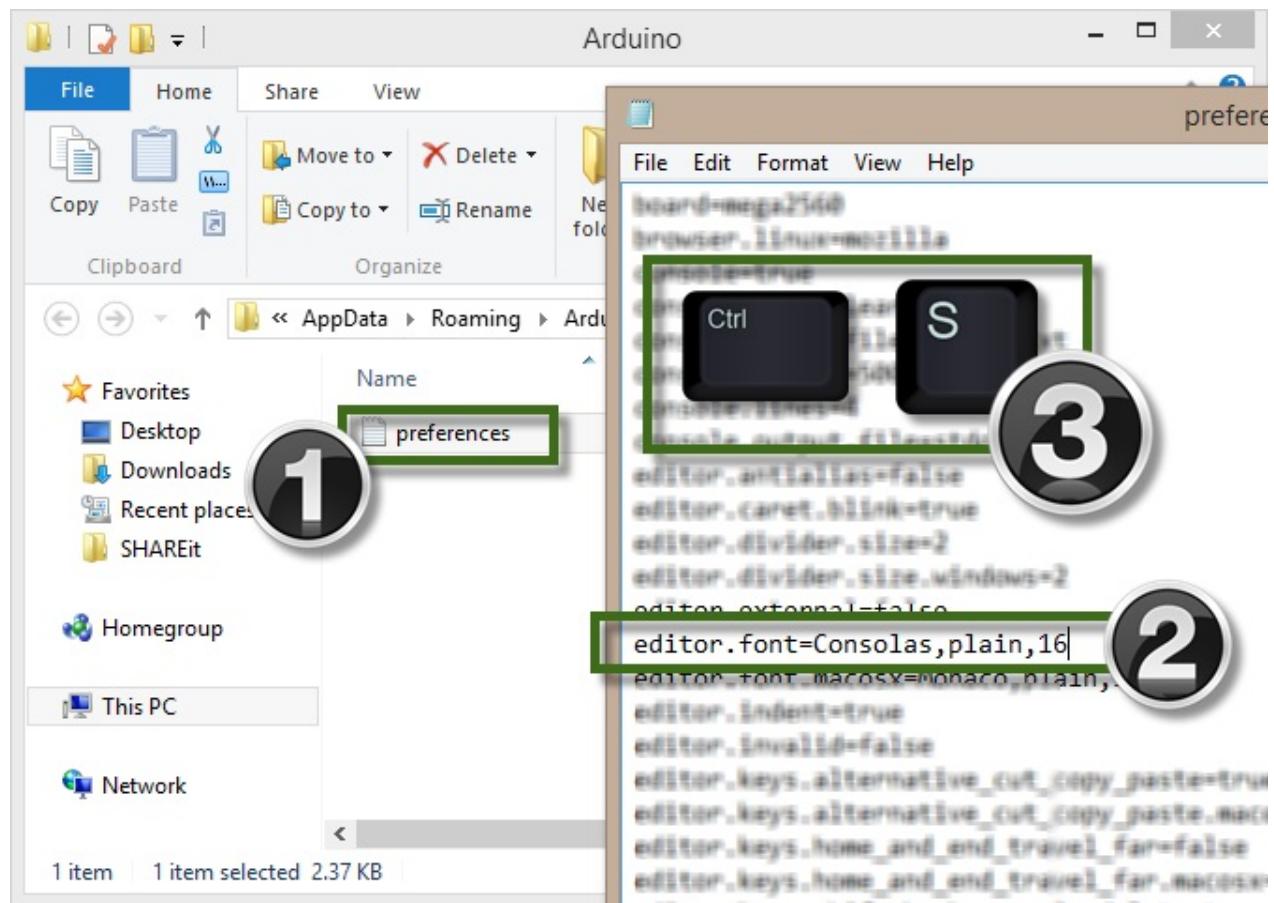


- তীব্র চিহ্নিত লেখাটায় ক্লিক করুন:

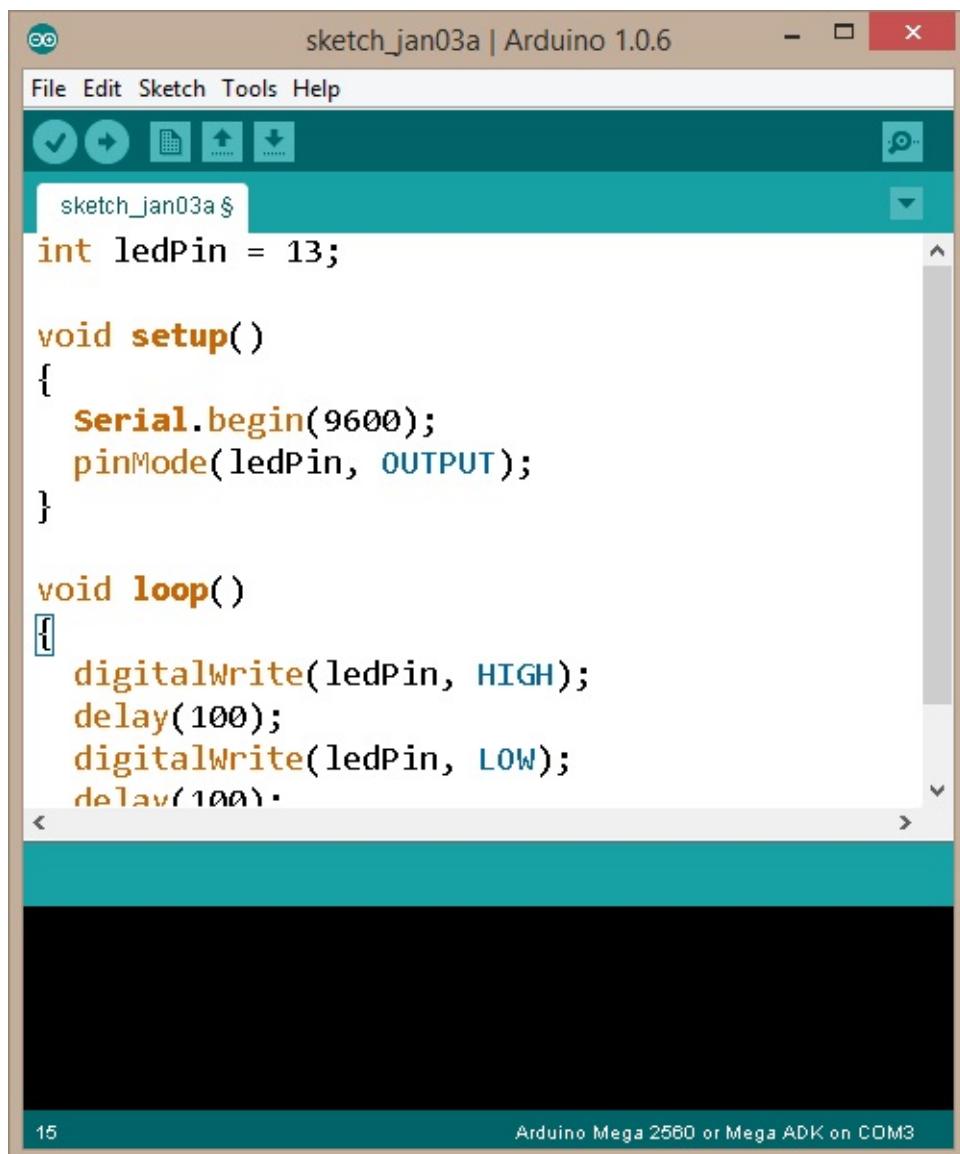


- নতুন ফোল্ডার আসবে একটি টেক্সট ফাইল দেখতে পাবেন preferences.txt নামে, সেটা ওপেন করুন, editor.font খুঁজে বের করে Monospaced এর জায়গায় আপনার পছন্দের ফট্টের নাম লিখুন [আমি এখানে উইঙ্গেজের সবচেয়ে ভাল Monospaced font হিসেবে Consolas ব্যবহার করেছি], plain কে পরিবর্তন না করে পাশে যে সংখ্যা আছে সেটা পরিবর্তন করুন। সংখ্যাটি ফট্টের সাইজ নির্দেশ করবে।

সবকিছু এডিট করার পর অবশ্যই **Ctrl+S** বা **Save** করবেন!



- এবার IDE Restart করলেই দেখবেন ফন্টের আকার ও ফন্ট পরিবর্তিত হয়েছে।



କିଛୁ ସାଧାରଣ ସମସ୍ୟା ଓ ତାର ସମାଧାନ [F.A.Q / Troubleshooting]

সমস্যা: আমার আড়ুইনো UNO এর ড্রাইভার কোনভাবেই স্টোপ দেওয়া যাচ্ছে না। সবগুলো পদ্ধতি ট্রাই করলাম, কুজ করছে না। আমি উইন্ডোজ ৮ / ৮.১ / ৭ ব্যবহার করছি।

সমাধান:

উইল্ডেজ ৮/৮.১ এর জন্য:

Shift কি চেপে আপনার পিসি রিস্টার্ট দিন। দেখবেন রিস্টার্ট নিয়ে একটা বু স্ক্রিন এসেছে। Choose an option আসবে, Troubleshoot > Advanced options > Startup settings > Restart। এবার পিসি রিস্টার্ট হলে আবার বু স্ক্রিন আসবে ও কিছু লেখা আসবে, কিবোর্ড থেকে 7 চাপুন। এখন পিসি অন হলে আবারও একই নিয়মে ডাইভার সেটাপ দেওয়ার চেষ্টা করুন। এবারও না হলে উইন্ডোজ ৭ এর সমাধান টাই করুন।

উইকেজ ৭ এর জন্য:

আড়ইনো বোর্ডটি কানেক্টেড রাখন এবং পৰ কাজগুলো কৰুন।

Device Manager > Unknown Device [আপনার আর্ডুইনো বোর্ডটি] > Update driver software > Browse My computer for driver software > Let me pick from a list of device drivers from my computer > Modems > বামপাশ থেকে Compaq ও ডানপাশের Models থেকে Richochet Wireless USB Model সিলেক্ট করুন [লিস্ট আপডেট হতে একটু টাইম দিন] > Yes > Installing দেখাবে। ইন্�স্টলড হলে Baud Rate 9600 করে দিন ও একটি COM পোর্ট সিলেক্ট করুন। COM পোর্ট সিলেক্ট করার সময় 50-60 এর মধ্যে সিলেক্ট করার চেষ্টা করবেন। এখন আর্ডুইনো আইডিই থেকে ওই কম পোর্ট সিলেক্ট করে কোড আপলোড করুন।

সমস্যা: কোড আপলোড দিতে গেলে এই এরর দেখাচ্ছে : avrdude: stk500_getsync(): not in sync: resp=0x00

সমাধান:

আপনার আর্ডুইনো বোর্ডটির COM পোর্ট পরিবর্তন করতে হবে। Device Manager > Ports > Arduino [Right Click] > Properties > Port setting > Advanced > Com Port Number > ইচ্ছামত সিলেক্ট করুন (অবশ্যই যেগুলো unused)। এবার পরিবর্তিত পোর্ট সিলেক্ট করে কোড আপলোড দেওয়ার চেষ্টা করুন।

সমস্যা: আমার সমস্যাটি এই লিস্টে নেই

সমাধান:

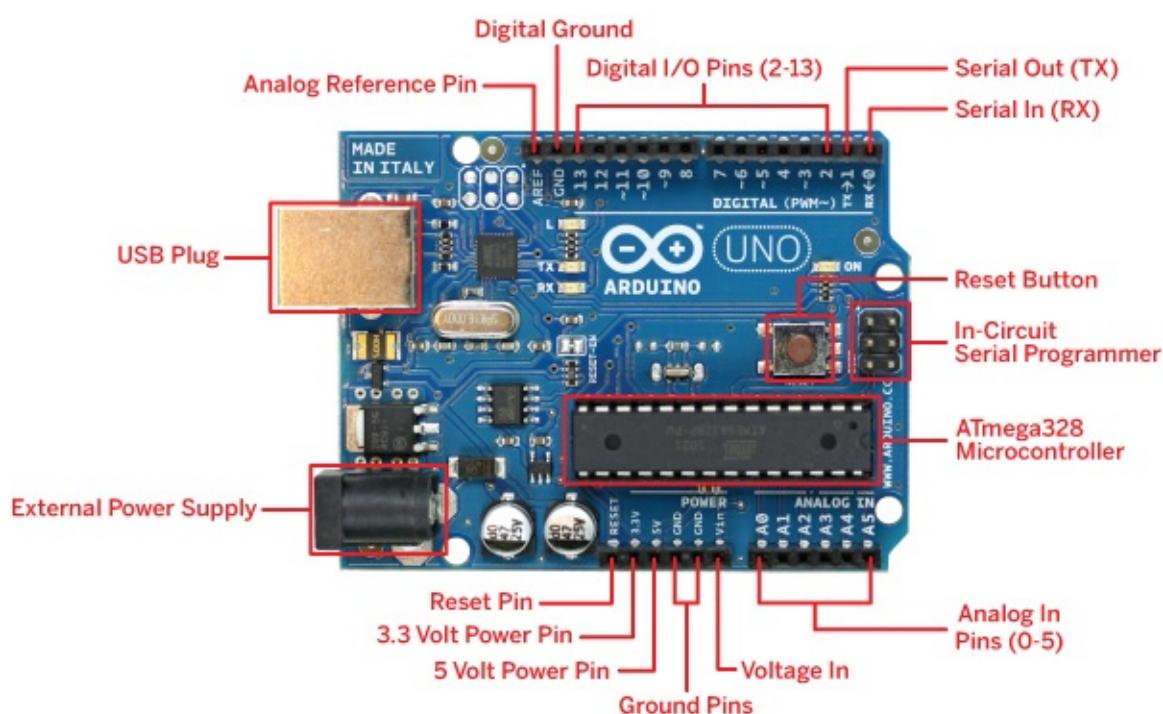
মন্তব্যে সমস্যাটি উল্লেখ করুন অথবা মেইল পাঠান manashmndi[অ্যাট]gmail[ডট কম] পরবর্তী পোস্টে আমরা আর্ডুইনো বোর্ডের ব্যাপারে বেশকিছু টার্ম ও কম্প্যানেক্ট সম্পর্কে জানব ও আর্ডুইনো প্রোগ্রামিংয়ের সাথে পরিচিত হব।

আডুইনো বোর্ড ও কোড সম্পর্কে ধারণা

যা যা লাগবে:

- একটি আডুইনো বোর্ড [Arduino Uno / Mega 2560]
- ব্রেডবোর্ড
- কিছু প্রিমিয়াম জাম্পার / 22AWG Solid Wires
- একটি LED
- একটি রেজিস্ট্যান [2200hm / 330 Ohm / 1k]
- সিম্যুলেশনের জন্য Proteus 8.0 SP1 or Latest Version

আডুইনো বোর্ড:



উপরের ছবিতে আডুইনো উনো বোর্ডের [Top-View] বিভিন্ন জিনিস দেখানো হয়েছে। বামপাশের উপরের থেকে শুরু করা যাক।

- USB Connector:** বামপাশ-উপরে-কোণায় যে স্টিলের মত জিনিসটি দেখা যাচ্ছে সেটি হল USB Connector, আপনার USB A-B ক্যাবলের A প্রান্ত Arduino র সাথে কানেক্ট করতে হবে। পিসির সাথে Arduino কানেক্ট করলে তার কাজের জন্য 5V সে পিসি থেকে নিয়ে নেয়।
- External Power Connector:** বামপাশ-নিচে-কোণায় কাল রংয়ের একটি কানেক্টর দেখবেন যেখানে 9-12V সাপ্লাই দিতে পারবেন। খবরদার AC সাপ্লাই দিবেন না :P।

- **Digital Operation Pin-outs:** বোর্ডের উপরে যে 17 টা পিন দেখতে পাচ্ছেন সেগুলোতে Digital Operation করা হয়। কিছু পিনের নিজস্ব বৈশিষ্ট্য আছে।
- **0, 1 :**এই পিন দুইটি Serial Communication এর জন্য। 0 পিনটি RX [Receive/ Receiver / Reception] এবং 1 পিনটি TX [Transmit/ Transmitter/ Transmission] [বিগিনারদের জন্য অনেক কথাই মাথার উপর দিয়ে যেতে পারে কিন্তু চিত্তার কিছু নেই এগুলো নিয়ে আলোচনা করা হবে]
- **2-13 :**এই পিনগুলো ডিজিটাল অপারেশনের জন্য। আউটপুট ডোল্টেজ 5V ইনপুটে 3V-5V পর্যন্ত। কিছু পিনের আগে ‘~’ চিহ্ন আছে, তারমানে ওই পিনগুলো দিয়ে PWM [Pulse Width Modulation] করা যাবে
- **GND :**এটা হল Digital Ground, যারা ইলেক্ট্রনিক্স সম্পর্কে পড়ালেখা করেছেন তাঁরা নিশ্চয়ই Ground এবং Digital Ground এর মধ্যে তফাত জানেন।
- **AREF :** Analog Reference Pin, Analog Signal ভালমত বিশ্লেষণের জন্য এই পিনটা লাগে। এই সিরিজে পিনটি ব্যবহৃত হবে না।
- **Reset Button:** বোর্ড ও বোর্ডের ডার্সন [Revision] ডেডে এই বাটনটির অবস্থান বিভিন্ন হতে পারে, যেমন Mega তে মাঝখানে, UNO R2 এ মাঝখানে ও R3 তে কোণায়। এই বাটনটির কাজ প্রোগ্রাম রিস্টার্ট করা। যারা ডাবছেন বাটনটি MCU থেকে প্রোগ্রাম ডিলিট করে দেয় তাদের ধারণা ভুল :P
- **In-Circuit Serial Programmer বা ICSP Header Pin [মেইন চিপের আশেপাশে]:** আপনি ইচ্ছা করলে একটি আডুইনো বোর্ড দিয়ে অন্য আডুইনো বোর্ড বা Atmega সিরিজের নির্দিষ্ট কিছু MCU প্রোগ্রাম করতে পারেন, অন্য MCU বা Board প্রোগ্রাম করার জন্য কিছু Specific পিন লাগে। পিনগুলো হল: MOSI {Master Out Slave In}, MISO {Master In Slave Out}, SCK {Serial Clock}, RESET, VCC {+5V}, GND {Ground}। কাজের সুবিধার্থে বোর্ডে পিনগুলো আলাদা করে দেওয়া আছে। আপনি ইচ্ছা করলে এই পিনগুলো ব্যবহার করতেও পারেন নাও পারেন।
- **Atmega 328P-PU MCU:** এটাই মূলত আডুইনোর প্রাণ। এটা হল Atmel Co. এর তৈরি AVR আর্কিটেকচারের ATmega সিরিজের মাইক্রোকন্ট্রোলার। এটাতেই মূলত আপনি আপনার কোডগুলো আধোড় করেন। এর যে Leg আছে সেগুলো আসলে বোর্ডের দুইপাশে যে পিনআউট আছে সেগুলো।
- **ATmega 16u2 / ATmega8u2 MCU:** সাধারণত এই চিপটি থাকে USB পোর্টের পিছনে বা আশেপাশে, এটি মূলচিপের মত তো বড় নয় বটেই, অনেক ছোট। এর কাজ হল Arduino এর সাথে Computer এর Serial Communication এ সহযোগিতা করা বা পিসির সাথে কথা বলা।
- **ICSP Header for ATmega 16u2 MCU:** আরেক সেট ICSP হেডার পিন দেখতে পাবেন ওই চিপের আশেপাশে। ATmega 16u2 বুটলোডার বার্ন অথবা অন্য চিপ ফ্ল্যাশ করার কাজে আপনি এই হেডার পিনগুলো ব্যবহার করতে পারবেন।
- **A0-A5 Analog In (pins):** সহজ কথায় সেন্সর কানেক্ট করতে এই পিনগুলো ব্যবহৃত হয়। UNO তে একসাথে 6 টা অ্যানালগ সেন্সর ব্যবহার করতে পারবেন। মেগার ক্ষেত্রে সংখ্যাটা প্রায় 16 টি। ADC [Analog to Digital Conversion] এর কাজে পিনগুলো ব্যবহার করা হয়। পিনগুলো 14, 15 ... 19 সংখ্যা দ্বারা ও প্রকাশ করা হয়ে থাকে।
- **Voltage in (Vin):** এই পিনে আপনি External Power সাপ্লাই করতে পারেন। এটিও 7-12V DC নিতে পারে।

- **GND:** এটা হল `Ground` পিন
- **3.3V:** অনেক মডিউলে 5V দিলে সেটি পুড়ে যেতে পারে তাই আডুইনো বোর্ড থেকে সেই সুবিধাও আপনি নিতে পারেন, এই পিন থেকে আপনি 3.3V DC আউটপুট পাবেন।
- **Reset:** যেখানে Reset পিন লাগবে সেটার জন্য এটা ধার নিতে পারেন :P

এছাড়াও আডুইনোতে ক্যাপাসিটর, 12V বেগুলেটর, বেজিস্ট্যাম, `16MHz` ফ্রিস্টাল, 3.3V বেগুলেটর, `RX-TX Led`, 13-pin LED, Power LED ইত্যাদি আছে।

আডুইনো প্রোগ্রামিং:

আডুইনো প্রোগ্রামিং সম্পর্কে ধারণার জন্য আমরা গতবারের করা `Blink` প্রোগ্রামটি আরেকবার দেখতে পারি। তার আগে বলে রাখা ভাল, আডুইনোর প্রোগ্রাম ফাইলের নাম `.ino` পূর্বে `.pde` ছিল।

```
int ledPin = 13; // Putting integer value in a variable [A single line comment]

/* This is a Multiline
comment */

void setup()
{
    pinMode(ledPin, OUTPUT);
}

void loop()
{
    digitalWrite(ledPin, HIGH);
    delay(1000);
    digitalWrite(ledPin, LOW);
    delay(1000);
}
```

যদের জাভা/সি/সি++ প্রোগ্রামিং ল্যাঙ্গুয়েজ সম্পর্কে ধারণা আছে তারা সহজেই কোডটি বুঝবেন। যদের প্রোগ্রামিং এক্সপেরিয়েন্স নেই তারা দয়া করে পুরোটা পড়বেন এবং যদের প্রোগ্রামিং এক্সপেরিয়েন্স আছে তাদের হাস্তা চোখ বুলিয়ে গেলেই হবে।

Line 1:

এখানে `int ledPin = 13` লেখা আছে। `int` হল ডেটাটাইপ যার পূর্ণ প্রকাশ `integer` [পূর্ণসংখ্যা]। `Variable` হল এমন এক জিনিস যার মান পরিবর্তিত হতে পারে। এই লাইনটির মানে হল `ledPin` নামের একটা ড্যারিয়েবল তৈরি কর আর তার ভেতর `13` সংখ্যাটি প্রবেশ করাও।

ভ্যারিয়েবল আমরা বক্স হিসেবে কল্পনা করতে পারি, ধরা যাক toy KhelnarBaksho = gari; এটা দ্বারা বুঝালাম toy হল একটা থিংটাইপ (thing) আর KhelnarBaksho একটি বক্স আর '=' চিহ্ন দিয়ে gari দিয়ে বুঝালাম আপনার খেলনার গাড়িটি এর মধ্যে রাখুন। (আবার toy কে সিরিয়াসলি কোন টাইপ ধরবেন না, এটা দ্বারা উদাহরণ দিলাম মাত্র :P)

নিচের কোডটি দেখা যাক:

```
toy khelnarBaksho = jeep;
khelnarBaksho = remoteControlledCar;
```

এই দুইলাইনের কোড আসলে কি বলতে চায়? প্রথমে আপনি একটি khelnarBaksho নামের বক্স তৈরি করলেন এবং তাতে আপনার খেলনা jeep রাখলেন। এবার ডারুন আপনি আপনার জন্মদিনে একটি remote_Controlled_Car পেলেন এবং সেটা আপনি আগের বক্সে রাখতে চান।

এখন যদি আমি, khelnarBaksho = remoteControlledCar লিখি তাহলে কিন্তু একটা ছেট সমস্যা আছে। সেটা হল এই কাজটি করার সাথে সাথে আপনার jeep টি replaced হয়ে যাবে এবং আপনি তাকে আর ফিরে পাবেন না। :P এখানে আরেকটি লক্ষণীয় বিষয় এই যে, আমি প্রথমবার toy কথাটা লিখেছি কিন্তু পরেরবার আর লিখিনি। আসলে প্রথমবার আমি যখন toy কথাটি লিখে বক্স তৈরি করেছি তখন অটোমেটিক তার জন্য মেমরিতে জায়গা তৈরি হয়ে গেছে যেটি একটি toy type জিনিস গ্রহণ করবে।

যদি আমি আমার jeep এবং remoteControlledCar দুইটাই রাখতে চাই তাহলে নিচের কোডটি লিখব:

```
toy jeepBox, remoteControlledCarBox;
jeepBox = jeep;
remoteControlledCarBox = remoteControlledCar;
```

ডেটাটাইপ প্রথম ভ্যারিয়েবল, ২য় ভ্যারিয়েবল, এভাবে ভ্যারিয়েবল তৈরি করা যায়। এবার দেখুন, জিপের জন্য এবং রিমোট কন্ট্রোল গাড়ির জন্য দুইটি বক্স তৈরি করেছি এবং একটাও হারানো যায় নি :P

'// দেওয়ার পরের কথাগুলোকে বলা হয় Comment, কমেন্ট করার কারণ হল, অনেক সময় যদি আপনি কোড লিখেন পরে বুঝতে পারবেন না কিসের জন্য লিখেছেন; সেটা বুঝার জন্য অথবা অন্যকে কোড বুঝানোর জন্য Comment অনেক সময় দরকার হয়। // দিয়ে সিঙ্গেল লাইন Comment করা হয়। কম্পাইলার যখন // দেখতে পায় তখন সে আর সেই লাইন কম্পাইল করার চেষ্টা করে না।

Line 3:

```
// এইটা দিয়ে মাল্টি লাইন কমেন্ট দেয়। কমেন্ট বিশাল আকৃতির হলে বাববার // দেওয়ার চেয়ে // / / ব্যবহার করা ভাল।  
:P
```

Line 6:

`void setup()`, প্রতিটা আডুইনো প্রোগ্রামে দুইটি ফাংশন থাকতেই হবে সেটা হল; `void setup() {}` এবং `void loop() {}`। ফাংশন হল সেই জিনিস যার মধ্যে কিছু স্টেটমেন্ট থাকবে। যেমন $y = f(x) = x^2$ একটি ফাংশন কাবণ $f(x)$ এ আর্গুমেন্ট* আমরা যা দিব x^2 স্টেটমেন্টটি কাজ করে আমাদের সেই সংখ্যার বর্গ আউটপুট দিবে। সি/সি++, জাভা, আরডুইনো ইত্যাদি প্রোগ্রামগুলোতে ফাংশনের শুরু শেষ বুঝাতে {} সেকেন্ড ব্র্যাকেট (Curly Braces) ব্যবহার করা হয়।

`void setup()` ফাংশনটি আডুইনো পাওয়ার আপ করার সাথে সাথে রান হয় এবং কেবলমাত্র একবার রান হয়। এই ফাংশনটি আমরা ব্যবহার করি যেকোনকিছু ইনিশিয়ালাইজ করতে, কোন পিন আউটপুট হবে না ইনপুট হবে সেটা নির্ধারণ করতে বা এমন কোন কাজে যেটা একবার করলেই হয়। সেই স্টেটমেন্টগুলো আমরা সাধারণত `void setup()` ব্যবহার করি। `void` বলতে বুঝাচ্ছে ফাংশনটি কিছু রিটার্ন* করবে না।

Line 8:

`pinMode(ledPin, OUTPUT);` এই স্টেটমেন্ট একটা ফাংশন দেখা যাচ্ছে যার নাম `pinMode(arg1, arg2)`। এই ফাংশনটির ডেফিনিশন বা কি কি কাজ, কীভাবে করবে সেটি আডুইনো লাইব্রেরিতে দেওয়া আছে। আমরা এভাবে লিখে ফাংশনটিকে Call** করলাম মাত্র। ফাংশনটির কাজ হল আডুইনো বোর্ডের কোন একটি পিনকে `INPUT` হিসেবে নিবে নাকি `OUTPUT` হিসেবে নিবে সেটা নির্ধারণ করা। `INPUT` বলতে সেন্সর, বাটন বা এইজাতীয় জিনিসপত্রকে বুঝায়, `OUTPUT` বলতে `Led, Motor` ইত্যাদি জিনিসগুলোকে বুঝায়।

`pinMode` ফাংশনটি দুইটা আর্গুমেন্ট নেয়। প্রথমটি `int` টাইপ বা আর পরেরটি `mode` টাইপ। `pinMode(int, mode)` বললে বুঝতে সুবিধা হবে। আডুইনোতে আমার জানামতে `mode` দুইরকম দেওয়া আছে `INPUT` আর `OUTPUT`। যেহেতু `Led` একটি আউটপুট তাই আমরা এটাকে `OUTPUT` হিসেবে ডিফাইন করলাম ফাংশনটির মাধ্যমে। আর এই প্রোগ্রামে আমার পিনটাকে মাত্র একবার বললেই হচ্ছে সেটা ইনপুট না আউটপুট হবে তাই আমরা ফাংশনটি `void setup()` এর মধ্যে `Call` করলাম। Simple as that.

Line 11:

`void loop() {}`, এই ফাংশনটির ভিতরে যতগুলো স্টেটমেন্ট থাকবে সেটা বারংবার রান হতে থাকবে [পাওয়ার যতক্ষণ আছে ততক্ষণ]। এটাকে সাধারণ কথায় একটা `infinity loop` এর উদাহরণ হিসেবে বলা যায়। আগেই বলা হয়েছে যেকোন আডুইনো প্রোগ্রামে আর কিছু থাকুক কি না থাকুক, `void setup` এবং `void loop` থাকতেই হবে। আপনি ইচ্ছা করলে `void setup` বা `void loop` এর ভিতরটা ফাকা রাখতে পারেন কিন্তু কোড কম্পাইল করার সময় আপনাকে এইদুইটা ফাংশন লিখতেই হবে।

Line 13:

`digitalWrite(ledPin, HIGH);` এইখানে আমরা আরেকটি ফাংশনকে `Call` করলাম যার নাম `digitalWrite(int, bool)`। এই ফাংশনটির কাজ হল একটি নির্দিষ্ট `Pin` এ 5V দেওয়া অথবা 0V দেওয়া। ফাংশনটির দুইটা আর্গুমেন্ট, একটি হল `int` আরেকটি হল `boolean` বা সংক্ষেপে `bool`। কম কথায়, `boolean` হল হ্যাঁ/ না জাতীয় লজিক। `HIGH` এর `Synonym` হিসেবে আমরা `true, 1, yes` ইত্যাদি ধরতে পারি এবং `LOW` এর `Synonym` হিসেবে `false, 0, no` হিসেবে ধরতে পারি।

ফাংশনটি আসলে যেটা করে সেটা হল, _____ নাম্বার পিনে ডোকেজ দাও/নাও। এই ফাংশন কিছু রিটার্ন করে না।

Line 14:

`delay(1000)`, ডিলে আবারও আডুইনোর বিস্ট-ইন একটি ফাংশন। `delay(int)` ফাংশনটির আগুমেন্টে যে সংখ্যা আপনি পাস করবেন সেটার একক হল মিলিসেকেন্ড। তারমানে `1000` পাস করলে `1s` বুঝায়। `delay` এর কাজ হল একটি নির্দিষ্ট সময় ধরে আডুইনো বোর্ডকে পুরা হ্যাঁ করে রাখা। [ফ্রেডিং এ অনেকটা `thread.sleep(int)` এর মত]

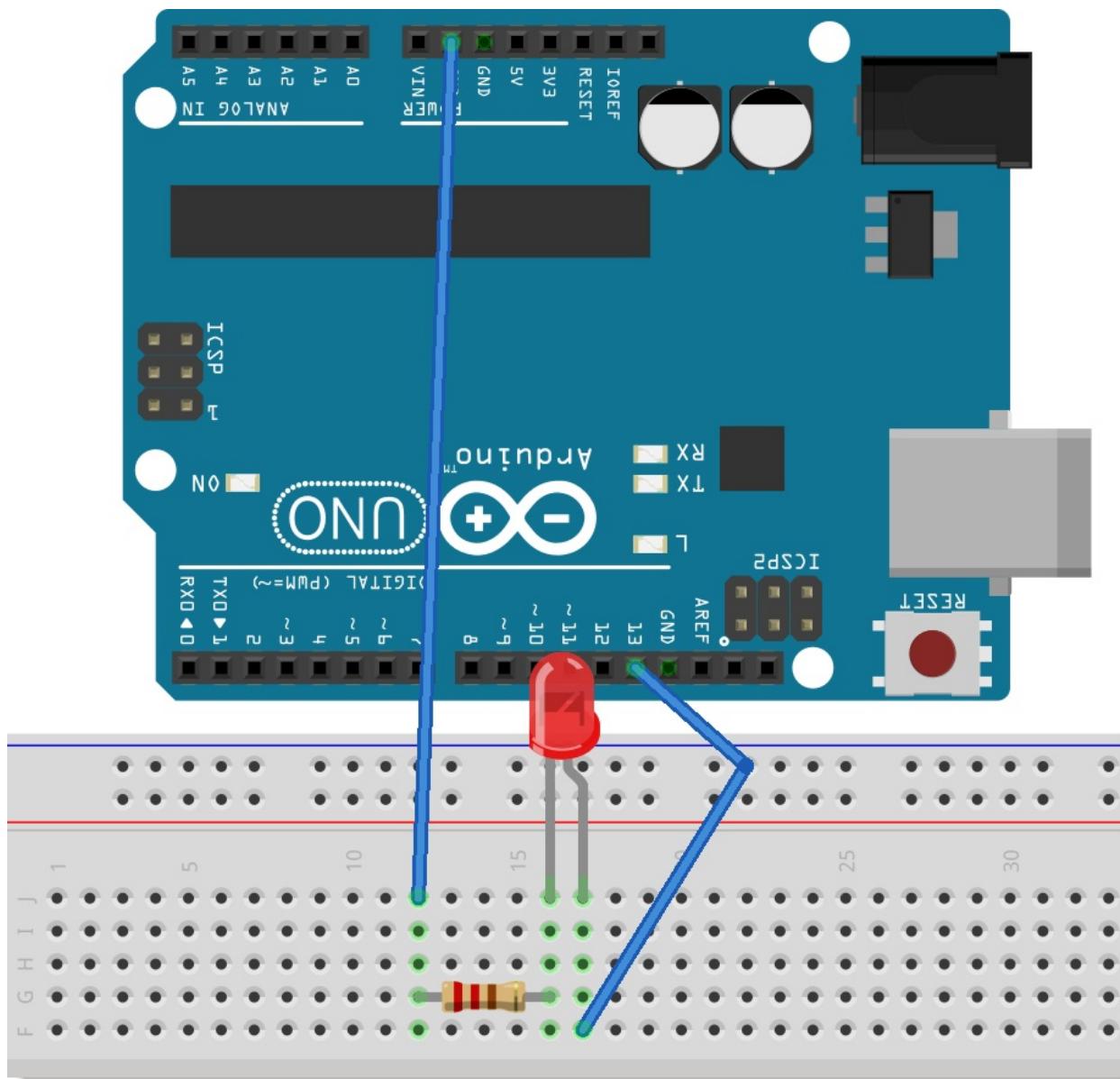
মনে রাখতে হবে `delay` তে যত সময় দেওয়া হবে সে ততক্ষণ পর্যন্ত পরবর্তী লাইন এক্সিকিউট করবে না এবং সেই কারণে তখন আডুইনো কোন কাজ-ই করবে না। ইনপুটের সময় ডিলে ফাংশন যতটা এডানো যায় তত ভাল।

একনজরে পুরো প্রোগ্রাম:

```
* একটি অ্যারিয়েবল ledPin নাম এবং তাতে 13 রংযাতি রংরূপ কর
* একবার চলবে { ledPin কে অটপুট হিয়েবে অট করো } [void setup এ]
* চলতে থাকবে {
*   ledPin এ ভোল্টেজ দাও
*   1000 মিলিয়েকেন্ড অপেক্ষা করো
*   ledPin এ ভোল্টেজ দিও না
*   1000 মিলিয়েকেন্ড অপেক্ষা করো
}
```

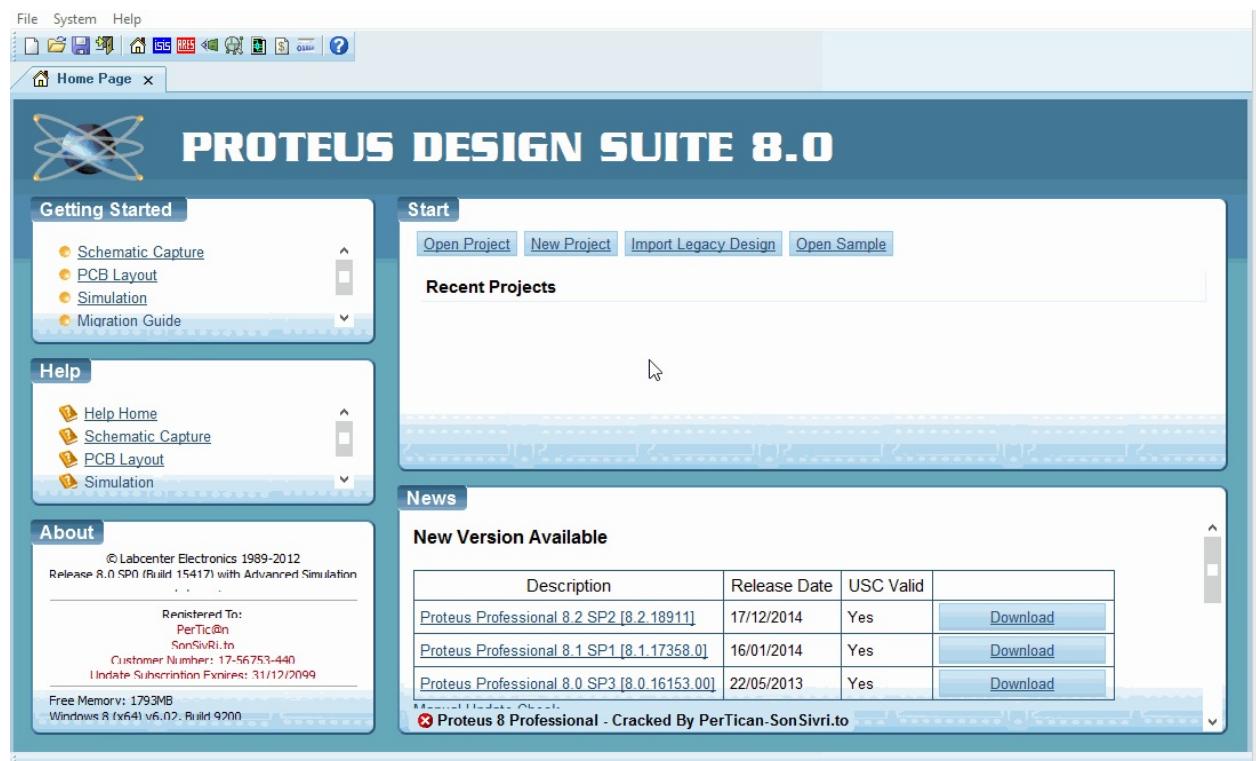
এভাবে আপনার কানেক্ট করা `led` টি জুলবে ও নিভবে। আশা করি প্রোগ্রামটি সবাই বুঝেছেন। না বুঝলে কমেন্ট অপশন তো থাকলাই।

সার্কিট ডায়াগ্রাম:

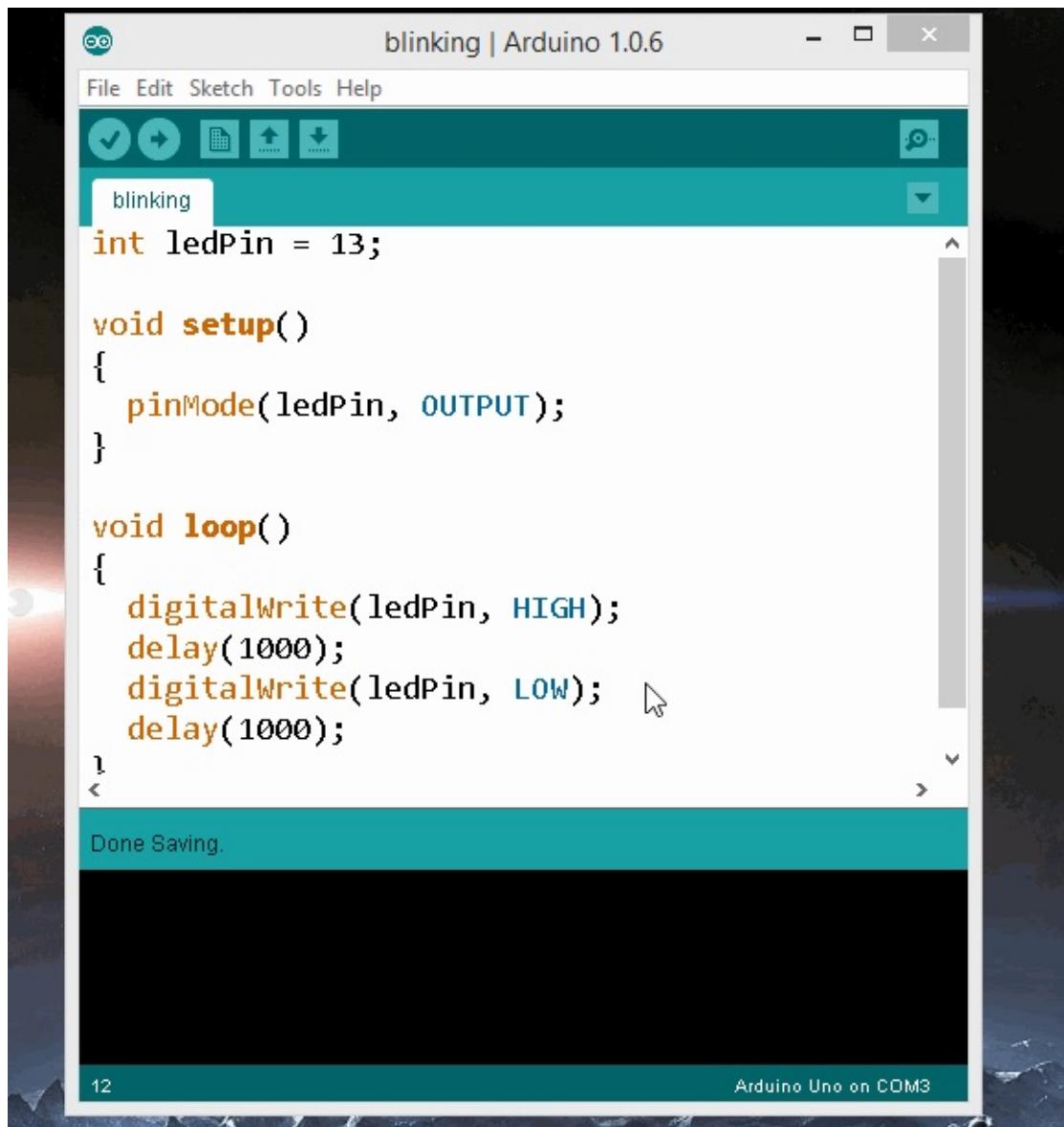


প্রোটিয়াসে সার্কিট কানেকশন ও সিম্যুলেশন (gif)

ISIS এ ডায়াগ্রাম আঁকা:



ভাৰ্যাল আড়ুইনোতে কোড আপ্লোড কৰা:



আপত্ত এই পর্যন্তই। পরবর্তী পোস্টে ডিজিটাল ইনপুট আউটপুট অপারেশন দেখানো হবে।

নোট:

প্রোগ্রামিংয়ের ফাংশন হবহু গণিতের ফাংশনের মতই। $y = f(x)$ যেকোন একটি ফাংশন হতে পারে কিন্তু $y = f(x) = 2x$ এ $2 * x$ স্টেটমেন্টের কাজ হল $f(x)$ এ আপনি যে ড্যালুটা পাঠাবেন এই স্টেটমেন্টের দ্বারা সেটাকে দ্বিগুণ করা হবে।

**ধৰা যাক, $z = f(r, x) = r^2 + x^2$, এখানে আমি আর্গুমেন্ট যদি দুটি সংখ্যা পাঠাই যেমন, $z = f(2,3) = 2^2 + 3^2 = 4 + 9 = 13$ হবে তার আউটপুট। ঠিক সেইভাবেই আমরা চিন্তা করতে পারি digitalWrite হল একটি ফাংশন যে আর্গুমেন্ট integer টাইপ ও bool টাইপ ডেটা নেয় আর একাধিক আর্গুমেন্ট থাকলে ',' দ্বারা লেখা হয়।

*আপনি যখন একটি ফাংশন তৈরি করলেন তখন অবশ্যই আপনাকে তৈরি করার সময় বলে দিতে হবে যে সেটা কি ফেরত [return] দিবে। ধৰা যাক, আপনার ওয়াশিংমেশিনটির মেথড washClothes, তাহলে আপনিই চিন্তা করুন; আপনার ওয়াশিংমেশিনয়ের মেথডের আর্গুমেন্ট কি এবং রিটুন টাইপ কি?

আপনি ওয়াশিংমেশিনয়ে কাপড় ইনপুট দেন এবং সেই কাপড়-ই আউটপুট পান (পরিষ্কার কাপড় অবশ্যই), তাহলে সেই হিসাবে ফাংশনটির ডেফিনিশন হবে cloth washClothes(cloth); ধৰুন আপনি মনে করুন আপনি আপনার ওয়াশিংমেশিন আপগ্রেড করতে গিয়ে তার মেথডটি পাল্টিয়ে void washClothes(cloth) করে দেন! এটা করার পরে দেখলেন ওয়াশিংমেশিনটি কাপড় কাঁচে ঠিকি কিন্তু সেই কাপড় আর ফেরত দিচ্ছে না :P। এই ঘটনার মূল কারণ হল তার রিটুন টাইপ আপনি পাল্টে দিয়েছেন, void এর সমার্থক শব্দ null বা নাই, তাই কাপড় কাঁচার পর সে আর কাপড় ফেরত দেয় না।

** ফাংশন Call, ধৰা যাক আপনি বিশাল অঞ্চল করছেন, কোন একজায়গায় আপনি দুইটা ফাংশন বানালেন, $f(x) = 5x$, $g(y) = 10y$ । এখন আপনি যদি অঞ্চল করার সময় কোথাও $f(2)$, $g(2)$ ইত্যাদি লিখেন তার মানে হল আপনি ওই ফাংশনটিকে Call করছেন কাজটি করে আউটপুট দেওয়ার জন্য। আডুইনো প্রোগ্রামিং আমরা কাজের সুবিধার্থে সব loop ফাংশনে না রেখে user defined function ব্যবহার করে ফাংশন তৈরি করে যখন দুরকার তখন Call করি।

সচরাচর জিজ্ঞাস্য প্রশ্ন: (F. A. Q)

প্রশ্ন:

আপনি blinking প্রোগ্রামটিতে ledPin নামক ড্যারিয়েবল নিলেন যেটার কোন দরকার দেখছি না, digitalWrite(13, HIGH) লিখলে কী কাজ করত না? আর আমার মনে হচ্ছে পিন নাম্বারকে প্রক্রিয়া হিসেবে ব্যবহার করা ডাল কারণ আমি তো আর পিন পাস্টাচ্ছি না।

উত্তর:

হ্যাঁ আপনার প্রতিটা কথাই ঠিক। `digitalWrite(13, HIGH)` লিখলেও কাজ করবে। মনে করুন, আপনার অনেকগুলো Led, Sensor ইত্যাদি দিয়ে একটি প্রজেক্ট বানালেন, এখন আমি যদি পিন নাম্বার দিয়ে কাজটি করতে যাই তাহলে আমার বারবার দেখতে হবে কত নাম্বার পিনে কোন জিনিসটি রেখেছি, সেই কাজটা সহজ করার জন্য আমরা `ledPin = 13` লিখলাম, এতে আমার 13 নাম্বার পিন না 1 নাম্বার পিন সেটা নিয়ে মাথা ঘামাতে হচ্ছে না। শুধু `ledPin` ড্যারিয়েবলটির কথা জানলেই হচ্ছে।

পরের অংশটাও ঠিক আমরা যেহেতু পিন নাম্বার পাঞ্চাছি না তাই কোন ড্যারিয়েবল কে ফ্র্যাক্ষন (constant) করার সিস্টেম হল ডাটাটাইপের আগে `const` শব্দটি লেখা। যেমন, `const int ledPin = 13;` এই স্টেটমেন্টের পর আমি যদি লিখি `ledPin = 14` তাহলে কোড কম্পাইল হবে না, কারণ আমি আগেই বলেছি `ledPin` একটি ফ্র্যাক্ষন আবার আমিই বলেছি `ledPin` এর মান পরিবর্তন করতে যেটা কম্পাইলার মেনে নিবে না।

প্রশ্ন:

Proteus এ আপনি .ino ফাইল না দিয়ে `blinking.cpp.hex` ফাইলটি দিলেন কেন? আর .hex ফাইল তৈরি করলেন কীভাবে?

উত্তর:

প্রোটিয়াস সিম্যুলেশনের জন্য বোর্ড বা মাইক্রোকন্ট্রোলার এ .hex ফাইল লোড করতে হয়, এছাড়া অন্য কোন ফাইল ফরম্যাট গ্রহণযোগ্য না। .hex ফাইল হল .ino ফাইলের কম্পাইল্ড ভার্সন। তাই আমি যখন Arduino IDE তে Verify বাটনে ক্লিক করি তখন .hex ফাইল তৈরি হয়।

প্রশ্ন:

.hex ফাইলটা খুঁজে বের করলেন কোন সফটওয়্যার দিয়ে?

উত্তর:

Everything নামক সফটওয়্যার দিয়ে। ফাইল খোঁজার দারুণ সফটওয়্যার। ডাউনলোড করুন [এখান](#) থেকে।

ডিজিটাল ইনপুট আউটপুট অপারেশন, **Loop** ও **If Else** এর উদাহরণ

এই পর্বে যা যা লাগবে:

- Arduino Board
- Breadboard
- Jumper Wire
- 10KOhm Resistor – ১টি
- 220Ohm Resistor – ১টি
- USB Cable - ১টি
- Pushbutton - ১টি
- Single-Color LED - ১টি

Loop (লুপ) ব্যবহার করে ডিজিটাল আউটপুটের উদাহরণ:

গত পর্বে আমরা দেখেছিলাম Led লিঙ্কিং কীভাবে করে। সেটাও কিন্তু ডিজিটাল আউটপুট। আমরা ডিজিটাল আউটপুট আরেকবার ঝালাই দেওয়ার জন্য For loop ব্যবহার করে দেখব কীভাবে ডিজিটাল আউটপুটের তৈরি করা যায়।

আর হ্যাঁ, আগের পর্বের সার্কিট ডায়াগ্রামটা ফলো করলেই হবে

প্রোগ্রামটা দেখে নেওয়া যাক:

```
int ledPin = 13;

void setup()
{
    pinMode(ledPin, OUTPUT);
}

void loop()
{
    for (int i = 100; i
```

Line 1, 3, 5, 8 এর ব্যাখ্যা পূর্বের পোস্টে দেওয়া আছে।

Line 10:

```
for (int i = 100; i <= 1000; i = i + 100){
// Statements goes here
}
```

এইখানে লুপ ব্যবহার করা হয়েছে। আমাদের অনেক সময় একই কাজ বারবার করতে হয়। একই কাজ বারবার করার প্রক্রিয়াই হল লুপ। যদি আমাকে বলা হয় C তে তোমার গ্রন্থের নাম ১০০ বার লিখ, সাধারণ চিন্তায় আমি দুইভাবে লিখতে পারি।

```
printf("Electroscholars"); // 1st one
printf("Electroscholars");
.....
.....
printf("Electroscholars"); // The 100th one
```

অথবা

```
int i;
for (i = 0; i < 100; i = i + 1)
{
    printf("Electroscholars");
}
```

এভাবে, আমাদের ১০০ লাইনের কোডটি মাত্র ৪ লাইনে সমাপ্ত হল। যদি লুপের স্টোকচার না বুঝে থাকেন তাহলে নিচের অংশটি পড়ে ফেলুন।

For লুপের স্টোকচার:

- `for` লুপের স্টোকচার শুরু হয় একজোড়া `First Bracket` দিয়ে, `First Bracket` এ ঠটি অংশ দুইটি `' ; '` – সেমিকোলন দ্বারা ভাগ করা থাকে
- প্রথম `' ; '` এর আগের অংশকে `Initialization`, প্রথম `' ; '` এর পরে এবং পরের `' ; '` এর আগের অংশকে `Condition` ও পরের `' ; '` এর পরের অংশকে `Increment` বলা হয়
- `first bracket` ক্লোজড করার পর `Second Bracket` বা `Curly Braces` থাকে। এই `Second Bracket` এর মধ্যে যেসব স্টেটমেন্ট / ফাংশন ইত্যাদি থাকবে তা `Loop` এর শর্ত ও ইনক্রিমেন্ট অনুযায়ী ততবার রান করানো হয়।

For লুপ যেভাবে কাজ করে:

- লুপটি প্রথমেই তাৰ ভ্যারিয়েবলে একটি মান বসিয়ে নেয় (`Initialization`) [উপরের উদাহরণে যেমন, `i = 0`]। দ্রষ্টব্য, C++, Java, Arduino ইত্যাদি ল্যাঙ্গুয়েজে প্রথম ব্র্যাকেটের ডিতৱ-ই ভ্যারিয়েবল তৈরি করা যায়, মানে `for (int i = 0; i < 100; i = i + 1)` Java, C++ ইত্যাদি ল্যাঙ্গুয়েজে কাজ করবে কিন্তু C তে নয়।

- এরপর সেই মানটি শর্তানুযায়ী চেক করে দেখে সেই ড্যারিয়েবল শর্তটি মানছে কিনা [Condition]
- যদি শর্ত মানে তাহলে সে এবাব কোন ‘Increment’ না করে সরাসরি Curly Braces এর স্টেটমেন্টগুলো রান করতে থাকে
- স্টেটমেন্টগুলো সব রান হয়ে গেলে এবাব সে তার ড্যারিয়েবল এব মান ইঙ্ক্রিমিনেন্ট শর্তানুযায়ী বাড়ায়।
- বাড়ানোর পরে আবাবও চেক করে সেটা শর্ত মানছে কিনা শর্ত মানলে সে লুপের স্টেটমেন্টে আবাব প্রবেশ করে রান করানোর জন্য ড্যারিয়েবলের মান যদি বাড়তে বাড়তে / কমতে কমতে এমন এক পর্যায়ে চলে যায় যেখানে সে আব শর্ত মানে না তখন আব লুপের স্টেটমেন্টগুলো রান করানোর চেষ্টা করে না

লুপের ব্যাপারটি আশা করি কিছুটা হলেও বোঝা গেল। এবাব দেখা যাক প্রোগ্রামটি কি বলছে?

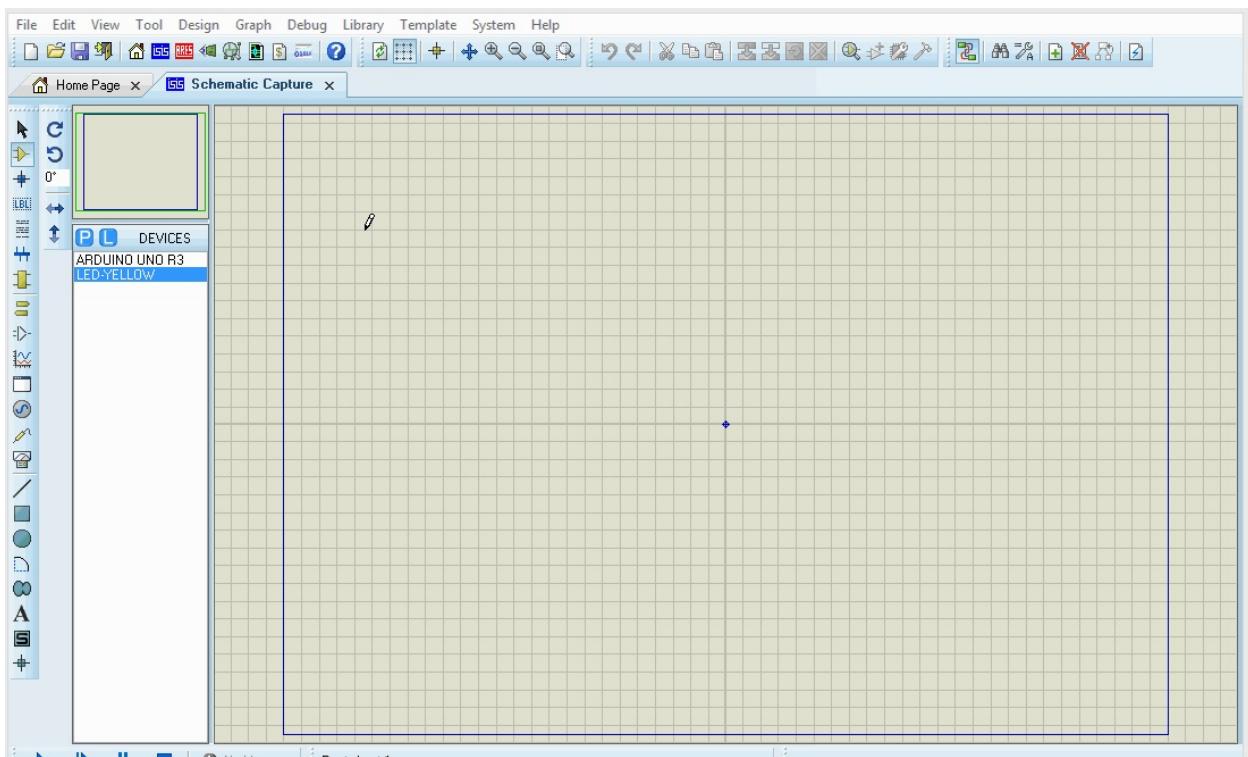
প্রোগ্রামটির Walkthrough!:

- `ledPin` নামের ইন্টিজার টাইপের ড্যারিয়েবল তৈরি করে 13 নাম্বারটি তার মধ্যে রাখ
- `void setup()` ফাংশনে: `ledPin` কে আউটপুট হিসেবে সেট কর
- `void loop()` ফাংশনে (এই ফাংশন একটি Infinite Loop এর উৎকৃষ্ট উদাহরণ):
- ফর লুপে `i` নামের ড্যারিয়েবল তৈরি করে 100 সংখ্যাটি তার মধ্যে রাখ, `i` কি 1000 এর চেয়ে ছোট? (যদি ছোট হয় তাহলে লুপে প্রবেশ করবে না হলে করবে না)
- হ্যাঁ
- লুপে প্রবেশ কর!
- `ledPin` এ 5 ডোর্ট দাও
- 100 মিলিসেকেন্ড অপেক্ষা কর
- `ledPin` থেকে 5 ডোর্ট সরাও
- 100 মিলিসেকেন্ড অপেক্ষা কর
- এবাব `i` এব মান আবও 100 বাড়াও ($=200$ হল)
- এখন `i` কি 1000 এর ছোট? হ্যাঁ, তাহলে লুপে প্রবেশ কর
- `ledPin` এ 5 ডোর্ট দাও
- 200 মিলিসেকেন্ড অপেক্ষা কর
- `ledPin` থেকে 5 ডোর্ট সরাও
- 200 মিলিসেকেন্ড অপেক্ষা কর
- এবাব `i` এব মান আবও 100 বাড়াও ($=300$ হল)

..... এভাবে `i` এব মান 1000 হলেও কাজ করবে কিন্তু যখন 1100 হয়ে যাবে তখন লুপে প্রবেশ করবে না, তখন সে `void loop` এব একদম প্রথমে চলে যাবে এবং তখন তাব কাছে এই `for` লুপটি নতুন লুপ মনে হবে তাই সে আবাব `i` এব মান 100 সেট করবে।

এভাবে যত সময় যাবে জুলা নেভার সময় বাড়তে থাকবে একসময় যখন `for` লুপের শর্ত মানবে না তখন `void loop` এব প্রথম থেকে চলা শুরু করবে।

Proteus সিম্যুলেশন:



ডিজিটাল Input Reading নিয়ে কিছু কথা:

ডিজিটাল ইনপুটের জন্য আগের সার্কিটটাকে একটু মডিফাই করে ব্যবহার করব। আমি যেটা করতে চাইছি সেটা হচ্ছে, একটি পুশ বাটন থাকবে আডুইনোর সাথে, সেটাকে প্রেস করলে একটি `led` জুলবে। খুবই সাধারণ একটি প্রজেক্ট কিন্তু ডিজিটাল রিডিং সম্পর্কে জানার জন্য যথেষ্ট। সার্কিটের জন্য কোড লেখার আগে কিছু জিনিস জানা জরুরি।

এখনে পুশবাটনটি কানেক্ট করলেই হবে না বরং তার সাথে একটি `Pull Down*` রেজিস্ট্যাম কানেক্ট করতে হবে। `Push Button` এর এক পাশে `5V` সাপ্লাই দেওয়া হয় আর আরেক পাশ আডুইনো বা অন্য কোন কম্পোনেন্টের সাথে যুক্ত করা হয়। যখন বাটনটি পুশ করা হয় তখন `5V` ও অপরপাশ `Short` হয়ে যায় তাই সরবরাহকৃত `5V` অপর প্রান্তে পৌছে যায়। এভাবে আমরা যখন বারংবার রিডিং চেক করে দেখব পিনে `5V` পাছে কিনা, সেই শর্ত দিয়ে আমরা আডুইনোর মাধ্যমে কাজ করতে পারি।

এখন আমরা চিন্তা করি, পুশবাটনটির সাথে কোন `Pull Down` রেজিস্ট্যাম নেই। তাহলে কি হবে? আপাত দৃষ্টিতে সমস্যা হওয়ার কথা না কারণ আডুইনো তো `5V` ট্লারেন্ট আর আমরা তার থেকে ভোল্টেজ নিয়ে তাকেই সাপ্লাই দিচ্ছি পুশবাটনটির মাধ্যমে, তাই আডুইনোর কিছু হবে না। কিন্তু যখন এক পাশ `5V` এর সাথে যুক্ত থাকবে `[Unpressed অবস্থায়]` এবং আমরা রিডিং চেক করব তখন `Unexpected Noise` এর সৃষ্টি হবে যেটা রিডিংকে `Fluctuate` করবে।

সমস্যা শুধু সেটাই না, আমরা ব্রেডবোর্ডে `5V Supply Rail` তৈরি করব যেখানে আমরা `5V` সাপ্লাই দিব, যখন `5V` দরকার হবে তখন সেখান থেকে `5V` নিয়ে কম্পোনেন্টে দিব। ধরা যাক, পুশবাটন আডুইনোর সাথে কানেক্টেড এবং পুশবাটনে `5V` সাপ্লাই দেওয়া ও `Push Button` এর একটি প্রান্ত গ্রাউন্ডে ও সেই গ্রাউন্ড থেকে একটি জাম্পার দিয়ে আডুইনোতে কানেক্ট করা হবে এবং একটি `Led` রেজিস্ট্যামসহ `13` নাম্বার পিনে কানেক্টেড। গ্রাউন্ড

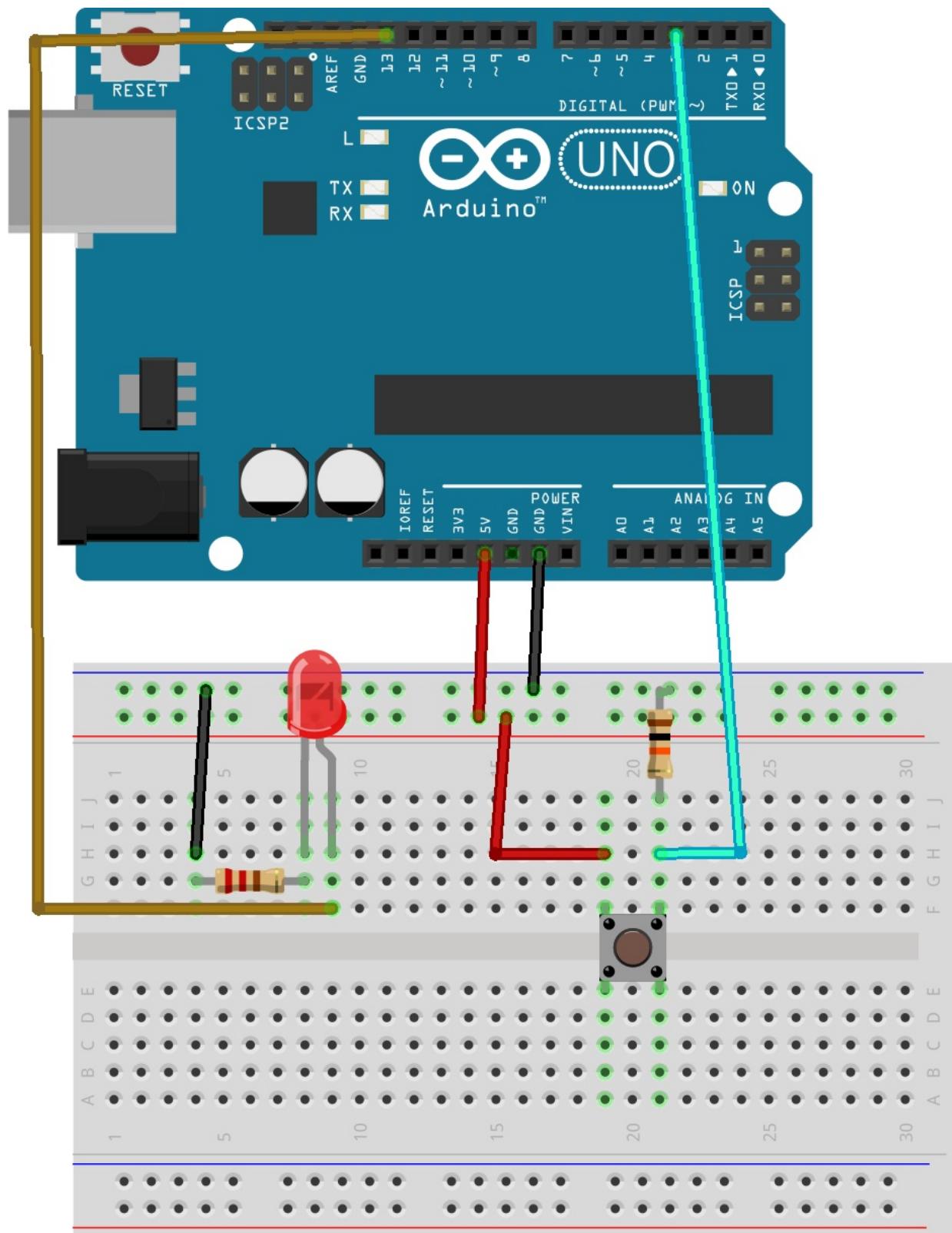
থেকে তাহলে দুটা কানেকশন যাচ্ছে, একটি সরাসরি গ্রাউন্ড অপরটি আড্রুইনোতে, পুশবাটন চাপ দিলে এবাব কারেন্ট কোথায় ফ্লো করবে? বেশিরভাগ কারেন্ট আড্রুইনোতে না গিয়ে গ্রাউন্ডে চলে যাবে অর্থাৎ রিডিং পাওয়ার পর্যাপ্ত কারেন্ট সে পাবে না। তাই আমরা একটি Pull Down রেজিস্ট্যাম ব্যবহার করছি।

Pull Down রেজিস্ট্যামের কারণে কারেন্টকে আবারও একই সিন্ধুতে নিতে হবে,

- হ্য সে হাই রেজিস্ট্যাম লাইন দিয়ে গ্রাউন্ডে যাবে
- অথবা অপেক্ষাকৃত অন্ন রেজিস্ট্যাম লাইন দিয়ে আড্রুইনোতে যাবে

0hm বলে গেছেন সে ২য় অপশনটি বেছে নিবে :P। তারমানে আমরা আড্রুইনোতে এবাব পর্যাপ্ত কারেন্ট পাচ্ছি রিডিং চেক করার জন্য। Resistance ব্যবহার করার কারণে কিছু ভোল্টেজ ড্রপ হবে অবশ্যই কিন্তু আড্রুইনোর Digital Reading এর জন্য পুরোপুরি 5V এর দরকার হ্য না :)

সার্কিট ডায়াগ্রাম:



আর্ডুইনো প্রোগ্রাম:

```

int led = 13;
int button = 3;

void setup()
{
    pinMode(led, OUTPUT);
    pinMode(button, INPUT);
}

void loop()
{
    if (digitalRead(button) == HIGH)
    {
        digitalWrite(led, HIGH);
    }
    else
    {
        digitalWrite(led, LOW);
    }
}

```

প্রোগ্রাম Walkthrough:

আগের লাইনের ব্যাখ্যা পূর্বের পোস্টগুলোতে দেওয়া আছে।

Line 7:

পুশবাটন থেকে আমরা রিডিং নিচ্ছি, তাই এটা ইনপুট। এখন এটি যে পিনের সাথে কানেক্টেড ঠিক সেই কারণে সেটাকে `Input` হিসেবে সেট করতে হবে।

Line 12:

`if else` অত্যন্ত গুরুত্বপূর্ণ একটি কনডিশনাল ফ্রে কট্টোল। এখানে যেটি করা হয়, প্রথমে আমরা একটি নির্দিষ্ট স্টেটমেন্ট সিলেক্ট করি। যদি সেটা `True` হয় তাহলে এই কাজটা করবে, সেটা `False` হলে আরেকটি কাজ করবে। এটা ঠিক করার জন্য `if else` দরকার।

```

if (booleanCheck == True)
{
    doThis();
} else {
    doThat();
}

```

যেমন, আরেকটি উদাহরণ:

```

if (6 < 7)
{
    printf("This is Electroscholars!");
} else {
    printf("Program will NEVER print this!");
}

```

6 সর্বদাই 7 এর চেয়ে ছোট, তারমানে `6 < 7 == True`। এর অর্থ প্রোগ্রামটি রান করলে “This is Electroscholars” প্রিন্ট হবে বাকি অংশ প্রিন্ট হবে না।

আমরা `if` এর মধ্যে নতুন একটি ফাংশন `digitalRead` দেখতে পাচ্ছি। `digitalRead` এর আর্গুমেন্ট ইন্টিজার টাইপ ও রিটার্ন টাইপ বুলিয়ান। অর্থাৎ সে আর্গুমেন্টে কোন পিন সেটা গ্রহণ করে এবং আউটপুটে `যাঁ/না, 1/0, HIGH, LOW` রিটার্ন করে।

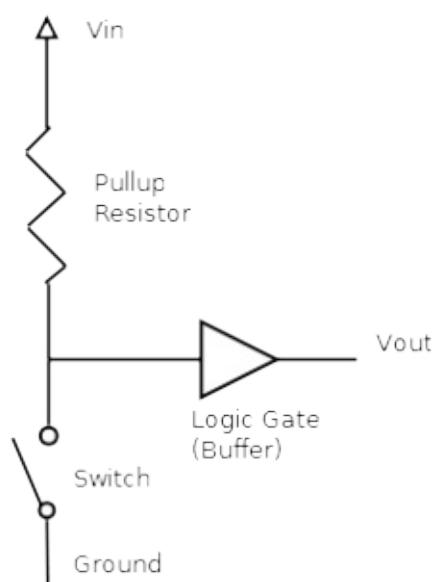
তাহলে যদি `button` Read করে `LOW` পায় অর্থাৎ ভোল্টেজ না পায় তাহলে সে `led` তে ভোল্টেজ দিবে না। বাটন প্রেস না করলে ভোল্টেজ পাবে না তাই আর `led` ও জুলবে না।

এখনে `else` এর আড়ালে আছে `if (digitalRead(button) == LOW)`, `if` এর পরে `else` থাকলে তার কন্ডিশন হ্য `if` এর উল্টা। অর্থাৎ, বাটনে থেকে `HIGH` রিড করলে `led` জুলবে।

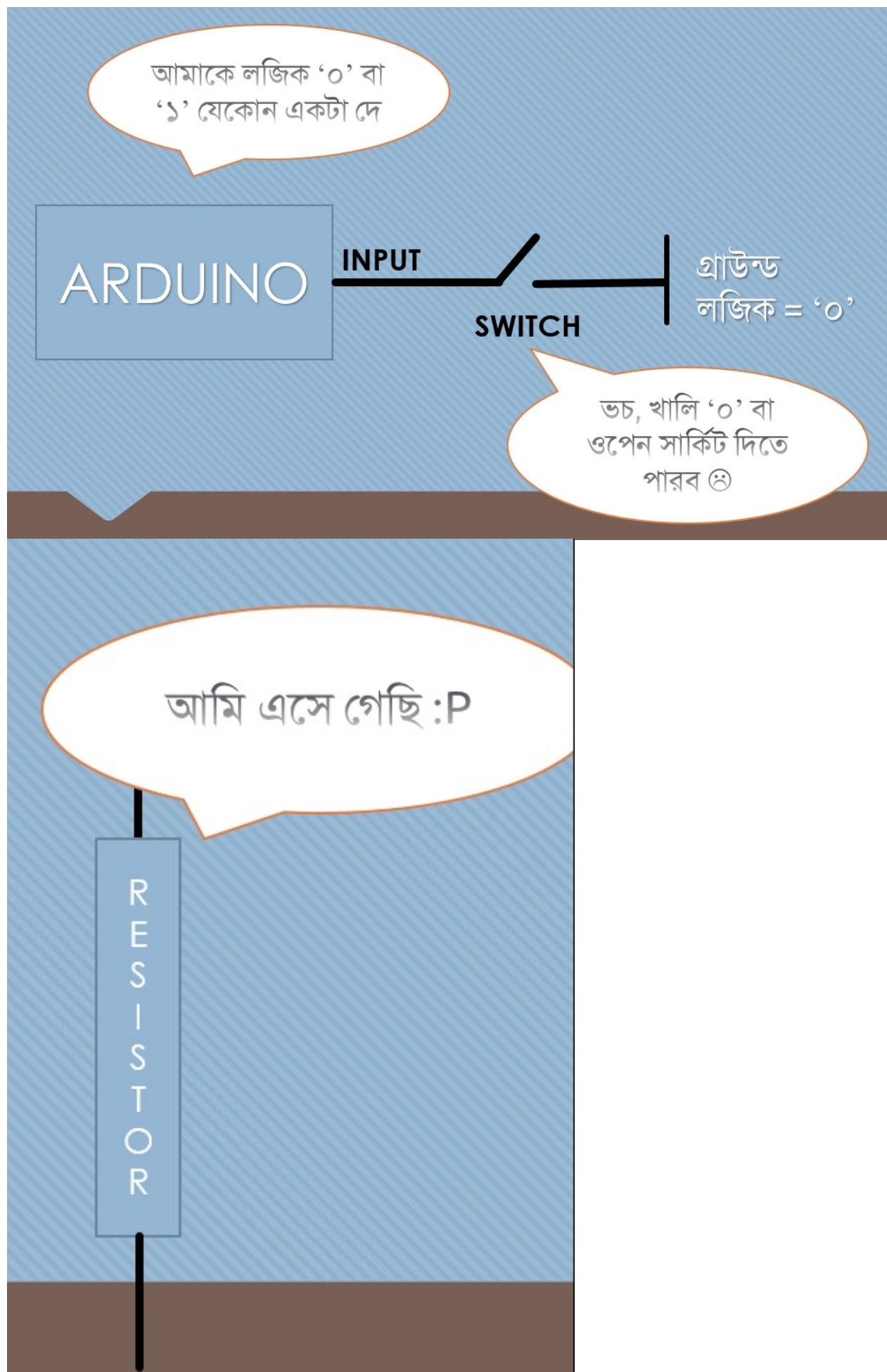
এটার সিম্যুলেশন করলে পুশ বাটন চাপ দিলে `led` জুলে ঠিকি কিন্তু আর সেটা নিন্দে না। এটার সিম্যুলেশনগত প্রবলেম আছে, তাই সিম্যুলেট করে দেখালাম না। এটা আপনাদের বাড়ির কাজ। বাস্তবে কোডটি কাজ করবে, সমস্যা নেই।

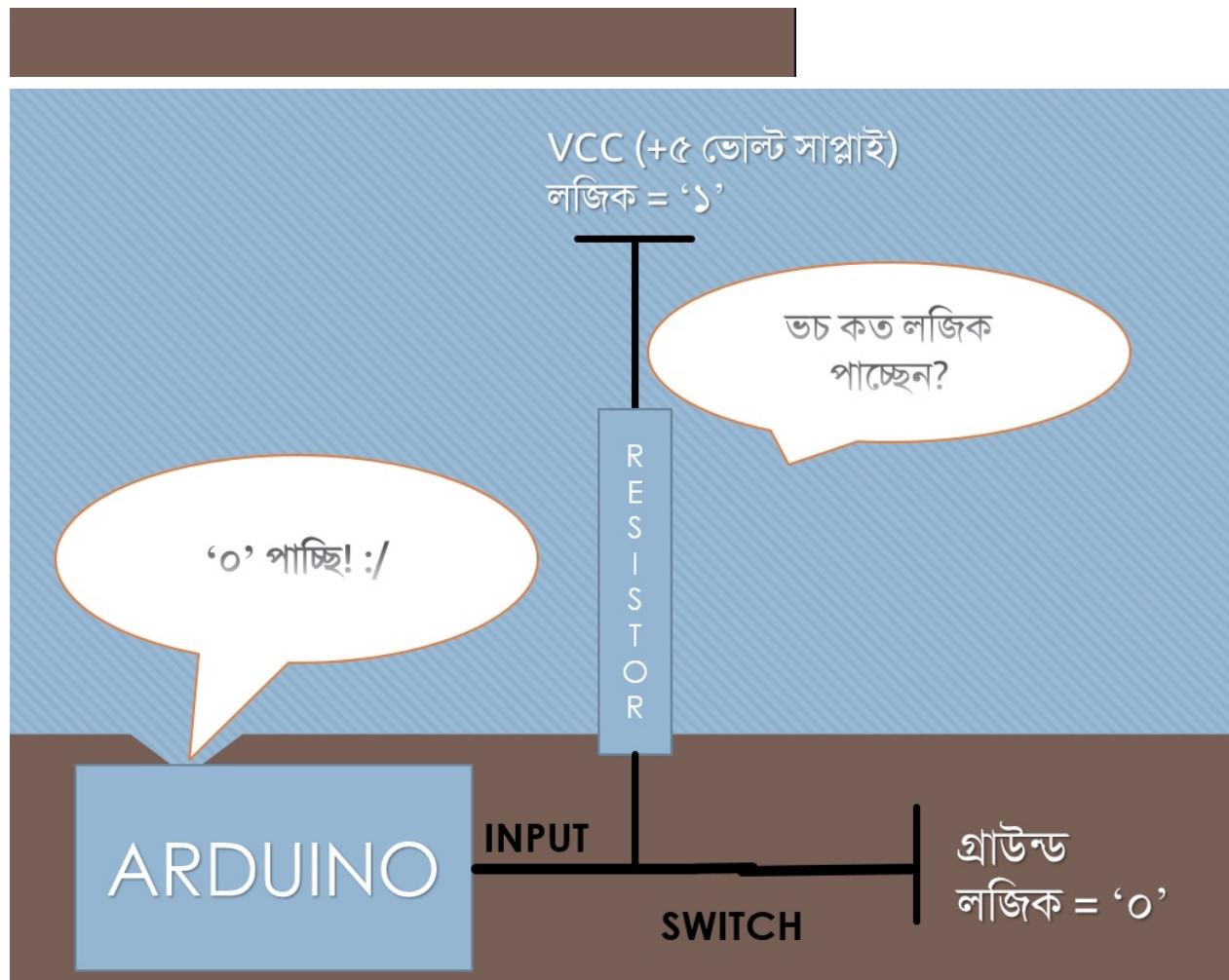
নোট:

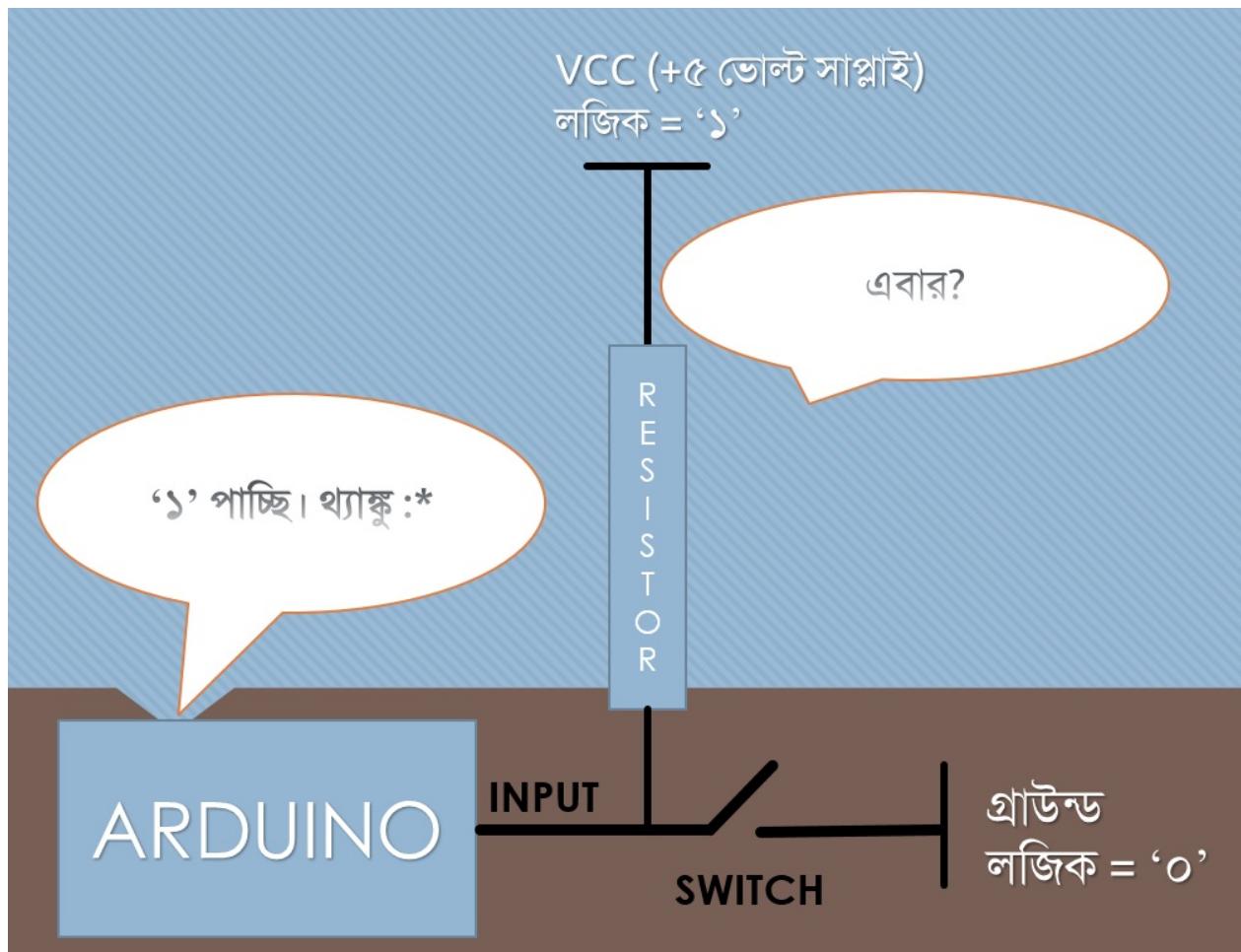
Pull Up Resistor, Pull Down Resistor



নিচের ছবিগুলো দেখলেই ধারণা পরিষ্কার হবে। Pull Down Resistor এর ক্ষেত্রে সুইচের সাথে গ্রাউন্ড না হয়ে VCC হবে।







আরও জানার জন্য ভিসিট দিন এই [লিঙ্কে](#)।

PWM (Pulse Width Modulation) পরিচিতি

যা যা লাগবে:

- Arduino [Predictable! Isn't it? :P]
- Breadboard
- Jumper
- Led
- Resistor [Recommended 100Ohm, but 220/330 Will work too]

PWM সম্পর্কে আলোচনা:

এতক্ষণে আপনারা ডিজিটাল ইনপুট আউটপুট দেখলেন, এবাব দেখাব অ্যানালগ আউটপুট কিভাবে আডুইনোর মাধ্যমে তৈরি করা যায়। ডিজিটাল আউটপুটের ক্ষেত্রে আমরা দেখেছি কোন পিনে আমরা `digitalWrite()` ফাংশনটি প্রয়োগ করলেই 5V যাচ্ছে। কিন্তু আমাদের অনেক সময় শুধু 5V দিলে হ্য না, কম ভোল্টেজ হলে সুবিধা হ্য। ডিজিটাল I/O Led রিংকিং কিংবা মোটর ঘুরানোর জন্য যথেষ্ট, কিন্তু মনে করুন আপনার মোটরটির স্পিড কন্ট্রোল কিংবা Led এর ব্রাইটনেস কন্ট্রোল করার দরকার হল, তাহলে কী করবেন? যেহেতু 5V অ্যাম্পাই করলে মোটরটি ফুল স্পিডে ও Led টি ফুল ব্রাইটনেসে জুলবে তাই তার জন্য একটা ব্যবস্থা নেওয়া জরুরি।

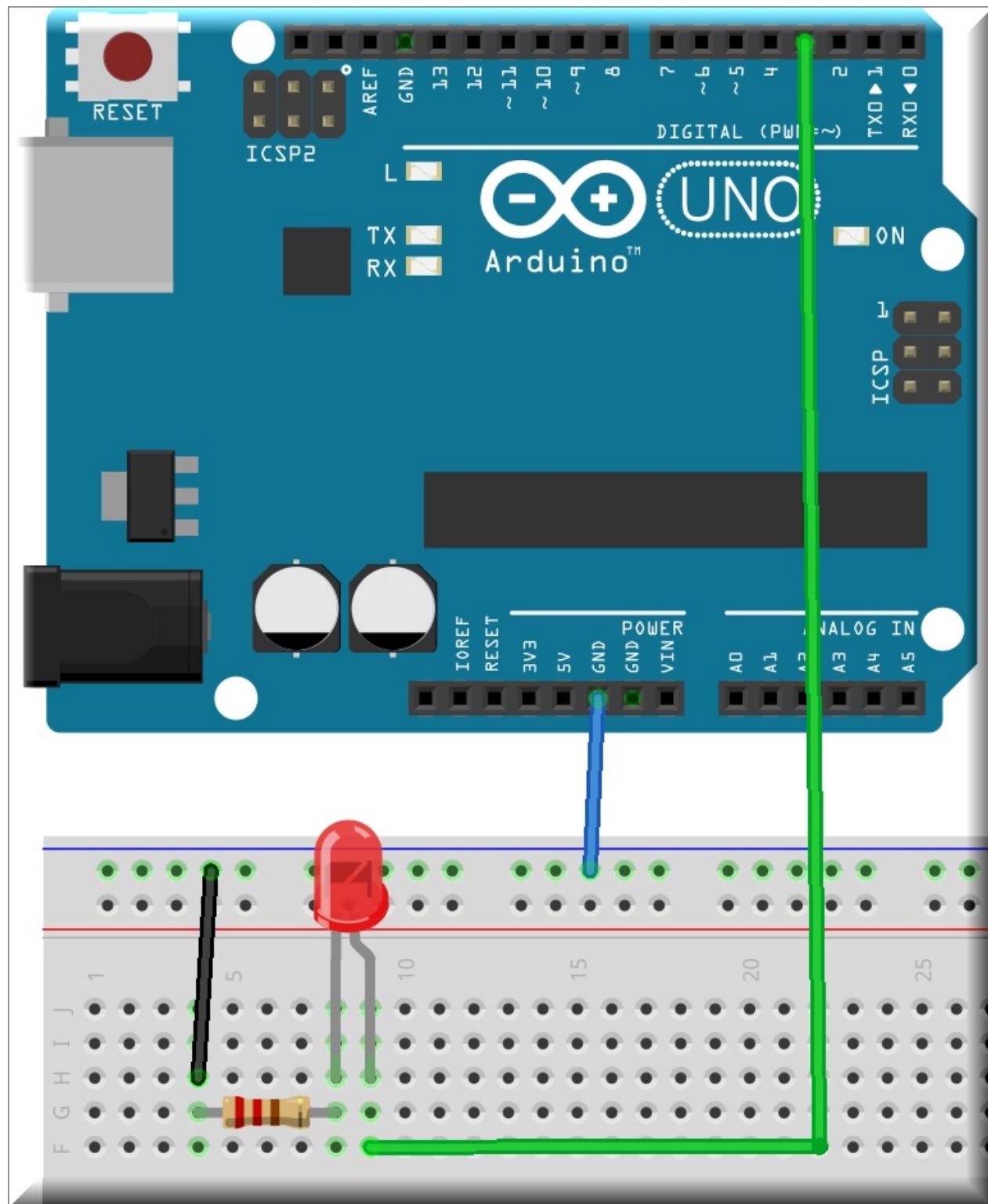
যখন আপনি এই সমস্যায় ডুগছেন, ঠিক তখনই PWM করা লাগবে। এটা দ্বারা ভোল্টেজ কমানো হ্য না, একটি সাইকেলে ভোল্টেজ কর্তৃক্ষণ দেব কি দেব না সেটা নির্ধারণ করাই PWM এর কাজ! PWM এর আরুক নাম `DAC` [Digital to Analog Conversion], এটি `DAC` থেকে `PWM` নামেই বেশি পরিচিত। যেহেতু `PWM` এর মাধ্যমে ভোল্টেজ আমরা টাইম ধৰে দেই ও দেই না, তাই এটা প্রায় অ্যানালগ আউটপুট দেয়।

`PWM` সিগনাল জেনারেট করার জন্য আমরা `analogWrite(int pin, int value)` ফাংশনটি ব্যবহার কৰি, `analogWrite` এর প্ৰথম আৰ্গমেন্ট `pin` নিৰ্দেশ কৰে কত নাম্বাৰ পিনে সিগনাল দিতে হবে এবং `value` নিৰ্দেশ কৰে কত ভ্যালুৰ `PWM` সিগনাল দিতে হবে। একটা কথা মাথায় রাখতে হবে, `value` এর মান 8-bit এর মধ্যে সীমাবদ্ধ রাখতে হবে বা 0 থকে $2^8 - 1$ বা 0 - 255 পৰ্যন্ত ভ্যালুৰ মান হতে পাৰে, এৰ চেয়ে বেশি হলে কাজ কৰবে না।

আৱ হ্যঁ, আডুইনোৰ সব পিন `PWM Signal` জেনারেট কৰতে পাৰে না, `UNO` এৰ ক্ষেত্রে যেসব পিনেৰ আগে ‘~’ চিহ্নটি আছে কেবল ওই পিনগুলোই `PWM Signal` জেনারেট কৰতে পাৰবে, যেমন 3, 5, 6, 9, 10, 11 পিনগুলো `UNO` এৰ `PWM` পিন। কোন মোটৰ স্পিড কন্ট্রোল বা `Led` এৰ ব্রাইটনেস কন্ট্রোলিংয়েৰ জন্য আপনাকে অবশ্যই সেই ডিডাইস/কম্পোনেন্টগুলো এই পিনগুলোতে কানেক্ট কৰতে হবে।

সার্কিট ডায়াগ্রাম:

সার্কিটটা হবহ তৃতীয় পরিচ্ছদের এর মত, তফাও হচ্ছে এখানে 3 নাম্বার পিনকে Led এর সাথে কানেক্ট করা হয়েছে।



প্রোগ্রাম:

```

const int led = 3;
void setup()
{
    pinMode(led, OUTPUT);
}
void loop()
{
    for (int i = 0; i < 256; i++){
        analogWrite(led, i);
        delay(10);
    }

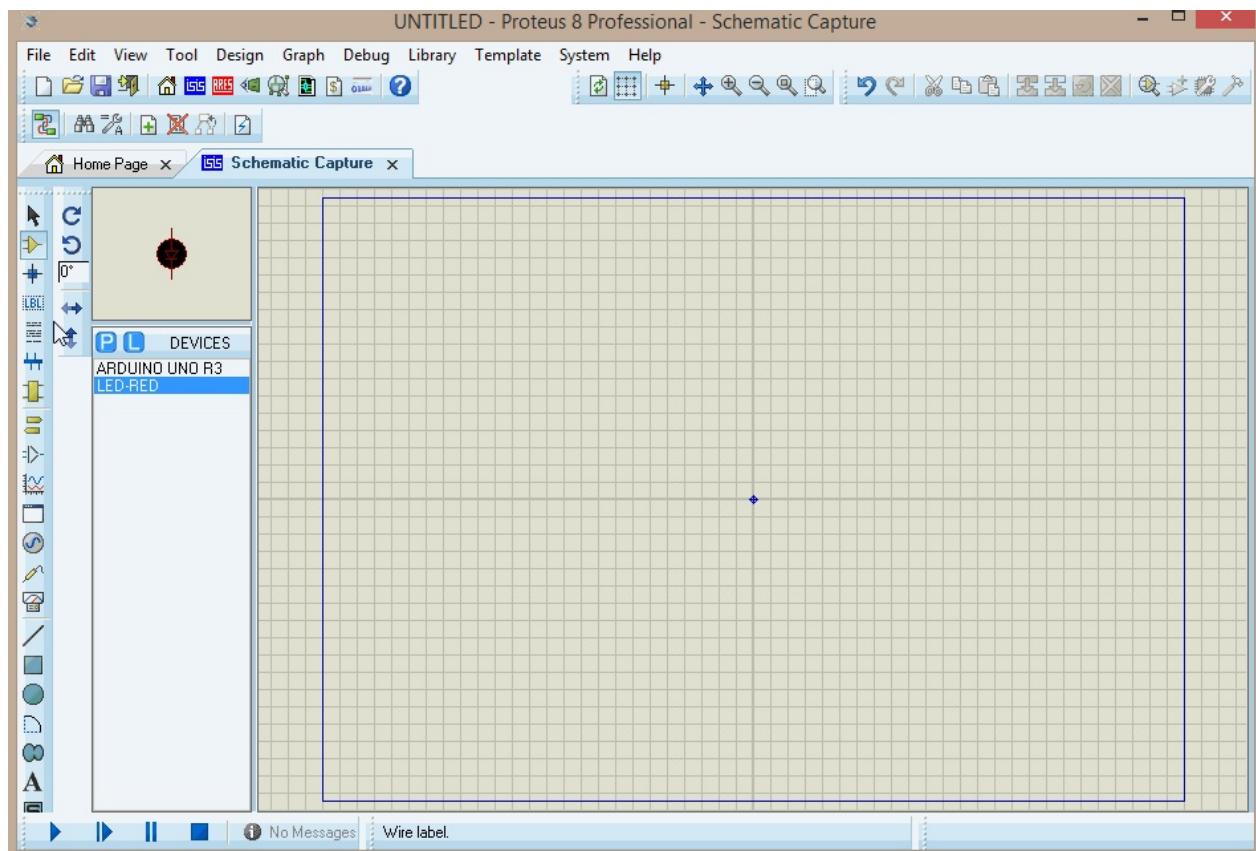
    for (int i = 255; i >= 0; i--){
        analogWrite(led, i);
        delay(10);
    }
}

```

প্রোগ্রাম Walkthrough:

- * একটি ইলুব টেরি কর led নামে এবং তার মধ্যে 3 রংখ্যাটি জ্বা কর
- * led কে আউটপুট হিমেরে চিনিত কর
- * ফর নূপ বানাও, নতুন ইন্টেজার লোকাল* অ্যারিয়েবল i কে Initialize করে 0 জ্বা কর; নূপটি তত্পর বান কর
- * led রে i এর অ্যান্টু অনুযায়ী analogWrite এর মাধ্যমে analog ফিল্ডাল জেনারেট কর
- * >o মিনিমেকেন্ড অপেক্ষা কর
- * i এর মান 1 বাড়, T3, i কি 256 এর ছোট? যদি আঁ- হয় তাহলে নূপে চুক; না হলে নূপে না চুকে পরের স্টে
- * i একটি লোকাল অ্যারিয়েবল নাও যার মধ্যে 255 জ্বা কর; i কি 0 এর মান বা বড়? আঁ- হলে নূপে প্রবেশ কর,
- * led রে i অ্যান্টু অনুযায়ী analogWrite এর মাধ্যমে analog ফিল্ডাল জেনারেট কর
- * >o মিনি মেকেন্ড অপেক্ষা কর
- * void loop() এর মধ্যকার জিনিস বারংবার চলতেই থাকবে

প্রোটিয়াস সিম্ব্যুলেশন:



নোট:

লোকাল ভ্যারিয়েবল ও প্রোবাল ভ্যারিয়েবল:

লোকাল ভ্যারিয়েবলগুলোকে যেকোন ফাংশন বা কোড ব্লকের মধ্যে ডিক্লেয়ার করা হয় এবং তার অস্তিত্ব থাকে শুধু ওই ব্লকের মধ্যেই। আলোচ্য প্রোগ্রামটিতে `i`, `i` হল লোকাল ভ্যারিয়েবল এবং `led` নামক ভ্যারিয়েবলটি প্রোবাল ভ্যারিয়েবল। আমরা জানি, প্রোগ্রামিংয়ে সাধারণত একই নামের জিনিসকে ২বার ডিক্লেয়ার করা যায় না। কিন্তু আমরা এই প্রোগ্রামে দেখতে পাচ্ছি `i` কে দুইবার ডিক্লেয়ার করা হয়েছে দুই লুপের ক্ষেত্রে। এটি সম্ভব হয়েছে শুধু একটা কারণেই, সেটা হল লুপ চালানোর পর কম্পিউটার তার মেমরিতে `i` নামের কিছুই জমা করে রাখেনি। লুপ শেষ হওয়ার সাথে সাথে সে `i` কে ভুলে যাচ্ছে। তাই `i` দুইবার ডিক্লেয়ার করা গেছে।

প্রোবাল ভ্যারিয়েবল কোন সাধারণত ফাংশনে থাকে না একে প্রোগ্রামের যেকোন জায়গায় ডিক্লেয়ার করা যায়। প্রোবাল ভ্যারিয়েবলগুলো একবার ডিক্লেয়ার করলেই হয়, দুই বা ততোধিকবার করতে গেলেই প্রোগ্রাম রান করবে না আমরা উপরের প্রোগ্রামটি `i` কে প্রোবাল হিসেবে ডিক্লেয়ার করেও কাজটি সম্পন্ন করতে পারি। যেমন:

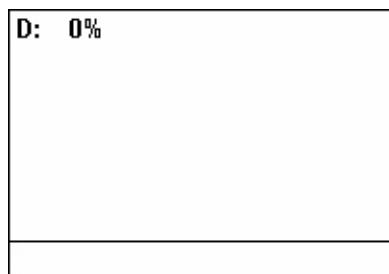
```
int i;
...
void loop()
{
    for (i = 0; i < 256; i++){ /* code */
        for (i = 255; i >= 0; i--) { /* code */
    }
}
```

Duty Cycle:

এটি সম্পর্কে Wikipedia কী বলছে?

A duty cycle is the percentage of one period in which a signal is active. A period is the time it takes for a signal to complete an on-and-off cycle.

অর্থাৎ, Duty Cycle হল এমন একটি টার্ম, কোন একটি সিগনালের ক্ষেত্রে Duty Cycle এর মান জানলে আপনি তৎক্ষণাত্মক বলে দিতে পারবেন একক পর্যায়কালে সিগনালটি কতক্ষণের জন্য on ছিল আর কতক্ষণের জন্য off ছিল। আমরা যেহেতু DC এর আলোচনা করছি তাই on মানে ধরে নেব 5V আছে এবং off মানে ধরে নেব 0V। নিচের এনিমেশন দেখলে ধারণা পাওয়ার কথা:



সূত্র: [উইকিপিডিয়া](#)

PWM এ আমরা আসলে কি করছি?

PWM করে আমরা আসলে Duty Cycle কন্ট্রোল করছি। যখন আমি `analogWrite(pin, 255)` আর্গুমেন্ট দিচ্ছি তার মানে এটি ওই পিনে 100% duty cycle এর সিগনাল দিবে তার মানে আমরা বলতে পারি, `digitalWrite(led, HIGH) == analogWrite(led, 255)`। কী? ব্যাপারটা Confusing লাগছে? কনফিউশনের দরকার নেই, এবার আপনিই ভাবুন, আমি যদি পর্যায়কালের ৫০% সময় ডোল্টেজ দিতে চাই ও বাকি ৫০% সময় ডোল্টেজ অফ রাখতে চাই তাহলে আমি `analogWrite` এ কত `Value` পাস করব? ঠিক ধরেছেন, `127`।

তার মানে, যদি ডোল্টেজের পর্যায়কাল `2 millisecond` হয় তাহলে ১ম মিলিসেকেন্ডে ৫ ডোল্ট থাকবে কিন্তু ২য় মিলিসেকেন্ডে ০ ডোল্ট থাকবে। আবার ৩য় মিলিসেকেন্ডে ডোল্টেজ থাকবে, ৪র্থ মিলিসেকেন্ডে থাকবে না..
....।

সচরাচর জিজ্ঞাস্য প্রশ্ন [F. A. Q]:

প্রশ্ন:

PWM যদি ডোল্টেজ না-ই কমায় তাহলে স্পিড বা ব্রাইটনেস কমে কীভাবে? এগুলো কী ডোল্টেজের উপর নির্ভরশীল নয়?

উত্তর:

PWM এ যত Low Duty Cycle হয় আমরা তত Dim Light পর্যবেক্ষণ করি। আসলে Dim Light এর জন্য দায়ী PWM নয়, আপনার চোখ! হ্যাঁ, Arduino এর Square Wave এর ফ্রিকোয়েন্সি প্রায় 490Hz এর মত। তার মানে, এটি প্রতি সেকেন্ডে 490 বার 5V থেকে 0V এ পরিবর্তিত হয়ে থাকে। আমরা জানি, $T = 1 / f$ । তারমানে, আডুইনোর পর্যায়কাল প্রায় 0.002 Second। এই কারণেই duty cycle কম দিলে dim দেখায় কারণ এটি এতটাই তাড়াতাড়ি ব্রিংকিং করে যে আপনার চোখ সেটা ধরতে পারে না। তারমানে আপনার চোখ আপনাকে বুঝায় যে led এর ব্রাইটনেস কমছে। আসলে led ফুল ব্রাইটনেসেই জুলে যা আপনি ধরতে পারেন না।

অবজেক্ট ওরিয়েন্টেড প্রোগ্রামিং পরিচিতি (প্রথম ভাগ)

যা যা দরকার:

- Codeblocks/Codelite/Eclipse বা সমানৈর IDE বা C++ কম্পাইলার, Text Editor – Syntax Highlighter সহ

আজকে Arduino নিয়ে তেমন কথা হবে না। শুধু OOP বা Object Oriented Programming নিয়েই আলোচনা করা হবে। OOP মূলত আলোচনা করা হবে C++ ল্যাঙ্গুয়েজ। C++ জানেন না, কোন সমস্যা নেই, যেকোন একটা ল্যাঙ্গুয়েজ OOP আয়ত্ত করলেই হয়, তবে ল্যাঙ্গুয়েজভেদে এর ইস্পিষ্টেশন আলাদা হয়। অবশ্যই OOP একটি বিশাল জিনিস কিন্তু এখানে মোটমাট ২ ভাগে OOP আলোচনা করা হবে কারণ কিছু করলেই কম কথায় OOP এর বেসিক জিনিসগুলো তুলে ধরা যাবে কিন্তু সেটা পরবর্তীতে সমস্যা তৈরি করতে পারে তাই সবদিক বিবেচনা করে OOP এর পোস্টটা যতটা পারা যায় Informative ও Implementation বেজড রাখার চেষ্টা করা হবে।

প্রোগ্রামিং মডেল [Programming Paradigm]:

আমরা যখনই কোন প্রোগ্রাম লিখি তখন নিচের মডেলগুলোর মধ্যে যেকোন একটি মডেলকে অনুসরণ করে প্রোগ্রাম লিখে থাকি। অনেক মডেলই আছে এবং কিছু কিছু প্রোগ্রামিং ল্যাঙ্গুয়েজ নির্দিষ্ট কিছু মডেল মেনে চলে। Java কে একটি পিওর OOP ল্যাঙ্গুয়েজ বলা চলে। কারণ Java কোড চালাতে গেলে আপনাকে কমপক্ষে একটি স্লাস ও একটি মেইন ফাংশন ব্যবহার করতেই হবে। আবার C++ এ আমরা Procedural/Functional/OOP ইত্যাদি যেকোন মডেল অ্যাপ্লাই করতে পারি। আমি এখানে দুইটি মডেলের কথা বলছি।

Procedural:

একে Imperative Programming বলা হয়ে থাকে। C তে আমরা যেসব কোড লিখি তা প্রায় সবই Procedural। কারণ কোডটি শুরু থেকে শেষ পর্যন্ত চলে এবং লাইন বাই লাইন (পড়ুন স্টেটমেন্ট বাই স্টেটমেন্ট) এক্সিকিউট হয়। অন্য কথায় Top To Bottom Approach। যখন একটা নির্দিষ্ট কাজ আমাদের করতে হয় এবং তার জন্য এমন একটি প্রোগ্রাম লিখলেই হয় যেটা টপ টু বটম রান করবে এবং সমস্যাটি সমাধান করবে তখন আমরা Procedural প্রোগ্রামিং মডেল ব্যবহার করে থাকি। এই মডেলে একাধিক ফাংশন ও একটি মেইন ফাংশন ব্যবহার করা হয়। BASIC, Pascal ও C তে Procedural Programming মডেল অনুসরণ করা হয়। সফটওয়্যার ডেভেলপিংয়ে এর প্রয়োগ নেই বললেই চলে। C তে একটি Procedural Programming এর উদাহরণ:

```

int add(int a, int b);

int main(){
    int firstNum = 6;
    int secondNum = 15;
    int sum;
    sum = add(firstNum, secondNum);
    printf("sum= ",sum);
    return 0;
}

int add(int a,int b){
    int result;
    result = a + b;
    return result;
}

```

Object Oriented:

একই প্রোগ্রাম যদি আমরা C++ এ অবজেক্ট ওরিয়েন্টেড স্টাইলে লিখি তাহলে হবে এইরকম:

```

#include <iostream>

class addNumbers
{
public:
    int sum;
    addNumbers(int, int); // এটা ইন Constructor, Constructor কিছু রিটার্ন করে না, চিন্তা করার দিন;
};

addNumbers::addNumbers(int num1, int num2)
{
    sum = num1 + num2;
}

int main()
{
    addNumbers object(1,2);
    std::cout << "Sum: " << object.sum;
}

```

Object Oriented Programming এর অ্যাপ্রোচ বটম টু টপ। অবজেক্ট ওরিয়েন্টেড ডিজাইনে যেটা থাকতেই হবে তা হল ক্লাস ও অবজেক্ট। এটা ছাড়া OOP চিন্তাই করা যায় না। বেশ কয়েকটি কারণে সফটওয়্যার ডেভেলপিং থেকে হার্ডওয়্যার প্রোগ্রামিংয়ে এর জনপ্রিয়তা চরম পর্যায়ের। OOP এর এই কারণ বা ফিচারগুলো হল:

- Class, Object
- Encapsulation
- Inheritance
- Polymorphism
- এগুলোর পাশাপাশি যেটা আলোচনা করা হবে: Constructor, Destructor

Class, Object:

আমাদের আশেপাশে যা আছে তাই Object এবং কতগুলো Object এর বৈশিষ্ট্য অনুযায়ী তাদেরকে একটা নির্দিষ্ট দলে ফেলা যায়। এই কনসেপ্টের উপর ভিত্তি করে Object গুলোকে একটি নির্দিষ্ট Class দ্বারা ডিফাইন করা হয়ে থাকে। আমরা গৃহপালিত পশুদেরকে একটি ক্লাস এর আওতায় আনি, নাম দিলাম Pet। C++ এ Pet কে ক্লাস হিসেবে ডিফাইন করতে হলে আমাদের লিখতে হবে class Pet। আচ্ছা, এবার Pet দের আমরা কি কি করি? তাদের নাম দিই, খাবার খেতে দিই এবং তাদের ধরণ অনুযায়ী আমাদের তারা Feedback দেয়। এই Feedback, Pet এর Object ভেদে বিভিন্ন ধরণের হতে পারে। কিন্তু ভালভাবে লক্ষ্য করলে দেখা যাবে তাদের কিছু সাধারণ বা Common বৈশিষ্ট্য ('Properties') আছে। যেমন তারা খাবার খায়, ডাক দেয়, নাম ধরে ডাকলে সাড়া দেয়, এবং ভাল ট্রেনিং পেলে অনেক কিছু করে দেখাতে পারে। তাহলে Pet ক্লাসের Object গুলোর Name, age ইত্যাদি হল তার Attributes, আর খাবার খাওয়া, ডাক দেওয়া ইত্যাদি কাজগুলো তার Method বা প্রোগ্রামিংয়ে আমরা Function আকারে লিখে থাকি। Method গুলোকে আবার Member Function হিসেবেও ডাকা হয়।

```
class Pet
{
public:
    string name;
    void eat();
    void utter();
};
```

আপাতত public: ** নিয়ে চিন্তা করার দরকার নাই। পরের লাইনগুলো দেখা যাক।

আমি name ড্যারিয়েবলকে string টাইপ হিসেবে ডিফাইন করলাম, তার Method গুলোকে ফাংশন হিসেবে ডিফাইন করলাম। যেটা মনে রাখতে হবে সেটা হল class Pet { // blablabla }; <- এই সেমিকোলনটার কথা।

আমি এখানে মেথডগুলোর একটা লিস্ট দেখিয়েছি। কিন্তু Method গুলো কীভাবে কাজ করবে সেটা লিখিনি। Method গুলো কীভাবে কাজ করবে সেটা দুইভাবে দেখানো যায়।

- ক্লাসের ভিতরে [Inside of Class]
- ক্লাসের বাহিরে [Outside of Class]

উপরের প্রোগ্রামটি যদি আমি ক্লাসের ভিতরে লিখে দেখাই তাহলে এমন হবে:

```
class Pet
{
public:
    string name;
    void eat()
    {
        cout << "Gulp gulp gulp..... . . ." << endl;
    }
    void utter()
    {
        cout << "meaw .... meaw ..." << endl;
    }
};
```

ঠিক যেমন আমরা ফাংশন ডিফাইন করি ওইভাবেই। এবার ক্লাসের বাইরে লিখলে দেখাবে:

```
class Pet
{
public:
    string name;
    void eat();
    void utter();
};

void Pet::eat()
{
    cout << "Gulp ... gulp... gulp.." << endl;
}

void Pet::utter()
{
    cout << "Meaw..... meaw.... " << endl;
}
```

প্রথমে আমরা একটা লিস্ট দিলাম কি কি থাকবে, তারপর Method গুলো কীভাবে কাজ করবে সেটা লিখলাম।
তাহলে, C++ এ ক্লাসের বাইরে Method ডিফাইন করার নিয়ম হল:

```
//[returnType] [ClassName] :: [methodName(argument)]
//Here-> '::' is called Scope Resolution Operator

void Pet :: eat()
{
    // Code goes Here
}
```

'::' কে স্কোপ রেজোল্যুশন অপারেটর বলা হয়। নামটা অনেক ভারী হলেও কাজটা সহজ। খালি বলে দেওয়া
কোন ফাংশন কোন ক্লাসে থাকে :)

এই পর্যন্ত যা শিখলাম:

- ক্লাস তৈরি করা
- ক্লাসের জন্য Attribute ও Method তৈরি করা
- Method কে ক্লাসের ভিতরে ও বাইরে কীভাবে ডিফাইন করা যায়

এখন যা যা শিখব:

- Object তৈরি করা
- Object এর Variable বা Attribute এ Value বসানো
- Object এর Method Call করা
- object কে Argument হিসেবে Pass করা
- Class, object ব্যবহার করে একটি পূর্ণাঙ্গ প্রোগ্রাম তৈরি করা

এই পর্যন্ত যেসব কিছু শেখা হয়ে গেল তা কি আসলেই শিখেছেন? না শিখলে কষ্ট করে আরেকটিবার পড়ুন, বুঝতে সমস্যা হলে Comment করুন। সামনে আরেকটু জটিল হতে পারে।

Object তৈরি:

```
class Pet {}; // Assume it is the class

int main()
{
    Pet dog; // Object Created!
}

// className objectName /* To create an object */
```

Object এর ড্যারিয়েবলে মান বসানো:

```

#include <iostream>
using namespace std;
class Pet
{
public:
    string name;
};

int main()
{
    Pet dog, cat;
    dog.name = "Tommy";
    cat.name = "Tom";
    cout << "Name of the cat is: " << cat.name << endl;
    cout << "Name of the dog is: " << dog.name << endl;
}

```

প্রোগ্রামে দেখা যাচ্ছে আমরা `Pet` ক্লাস দিয়ে দুইটি অবজেক্ট তৈরি করেছি। ধরলাম, একটা হল `Dog` আরেকটা হল `cat`। পোষাপ্রাণীর নাম রাখি বাঞ্ছনীয় তাই `dog` এর নাম দিলাম `Tommy` এবং `cat` এর নাম দিলাম `Tom`। এখনে আরেকটি ব্যাপার লক্ষ্যণীয়, আমরা নাম দেওয়ার সময় `dot operator` বা `objectName.variable = "Value"`। Scope Resolution Operator এর কাজ হল কোন Member Function কোন Class এর জন্তুর জন্য আর Dot operator এর কাজ হল Object এর নির্দিষ্ট কোন Member Function বা Variable এ Access কর্তব্য। `name` একটি `string` টাইপ জ্যাবিয়েবল Pet ক্লাসের জন্য কিন্তু মজার বিষয় হল Pet ক্লাসে আমি দুইটি Object' তৈরি করে তাদের ডি঱্রি নাম দিয়েছি যদিও তারা একইক্লাসের অস্তর্ভুক্ত।

ধরি, `variable = cat.name`, তাহলে `cat.name = "Tom"` হবে, `variable = "Tom"`, মানে `"Tom"` স্ট্রিংকে আমরা স্ট্রিংটাইপ ড্যারিয়েবলে বসালাম (অবশ্যই `cat` অবজেক্টের `variable` এ)।

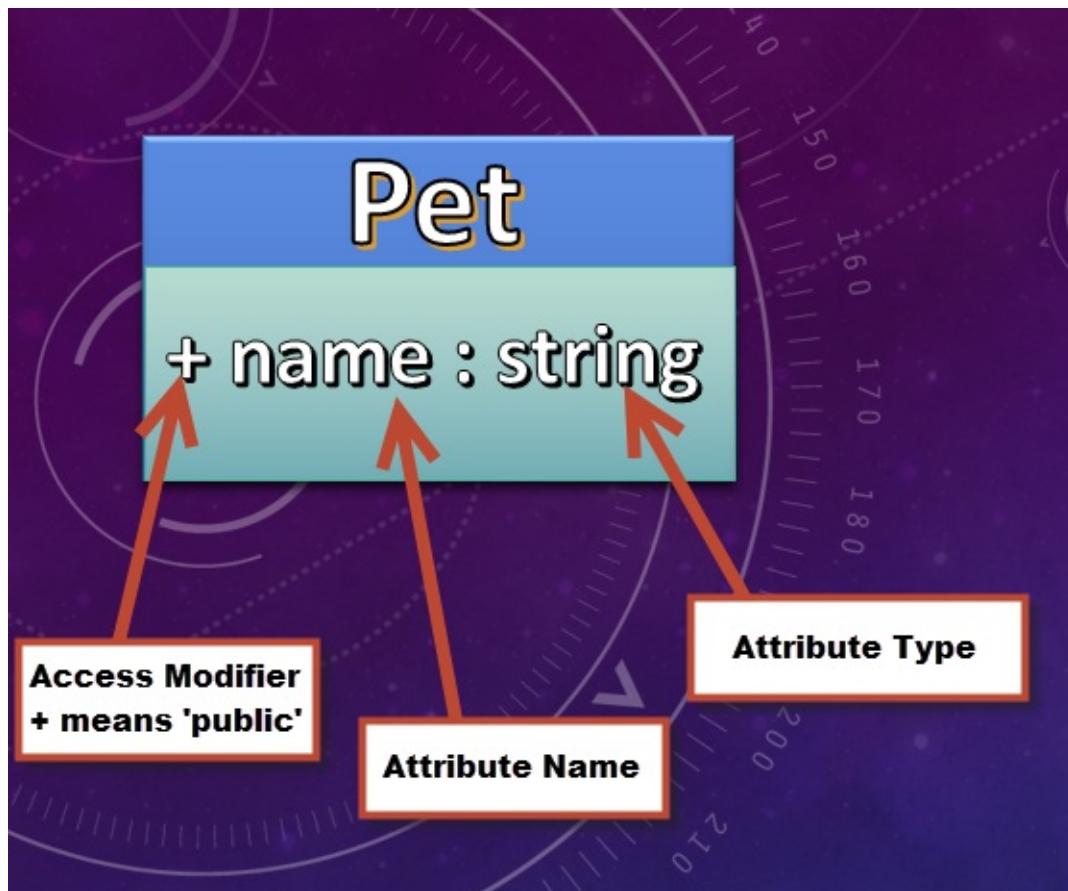
উপরের প্রোগ্রামের আউটপুট:

```

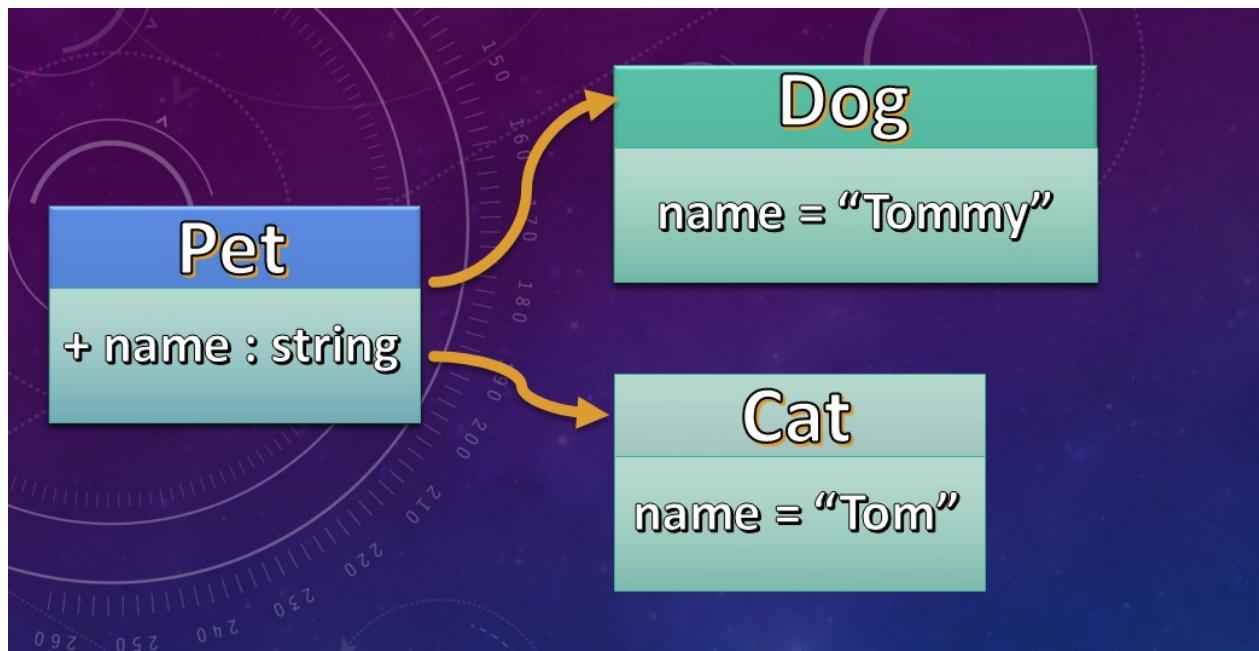
Name of the cat is: Tom
Name of the dog is: Tommy

```

নিচের ছবিটি `Pet` ক্লাসের **UML* (Unified Modeling Language) Diagram:**



আর এই ছবিটি হল অবজেক্ট তৈরি করার পরে:



Object এর Method Call করা:

আগের মতই, `object.functionName(argument)` এভাবে অবজেক্টের Method কে Call করা যায়।

```
#include <iostream>
using namespace std;
class Pet
{
public:
    string name;
    void sayName()
    {
        cout << "My name is " << name << endl;
    }
};

int main()
{
    Pet cat;
    cat.name = "Thomas";
    cat.sayName();
}
```

প্রোগ্রামে প্রথমে `cat Object` এর নাম অ্যাসাইন করলাম। এবং অ্যাসাইন করার পর `sayName()` ফাংশনটি
call করলাম।

আউটপুট:

```
My name is Thomas
```

কোন একটি ফাংশনে **Object** কে **Argument** হিসেবে পাস করা:

```

#include <iostream>
using namespace std;
class Pet
{
public:
    string name;
};

void sayName(Pet petObject, string petName) // This is NOT A MEMBER FUNCTION, it is just
{
    petObject.name = petName; // Name has been assigned by taking the name from the argument
    cout << "Name of the pet is: " << petObject.name << endl;
}

int main()
{
    Pet cat;
    sayName(cat, "Thomas"); // This will print "Name of the pet is: Thomas"
    cout << cat.name << endl; // This will print nothing!!!
}

```

আমরা জানি, ফাংশনে আর্গুমেন্ট পাস করা যায় দুইভাবে, Pass by reference এবং Pass by value হিসেবে। উপরেরটা হল Pass by Value। প্রোগ্রামে void sayName ফাংশনটি কোন ফ্লাসের অস্তর্ভুক্ত নয়, এটি একটি অবজেক্ট ও একটি স্ট্রিংকে আর্গুমেন্ট হিসেবে নিয়ে, অবজেক্টের Attribute name এ শুধু string টা বসিয়ে দিয়েছে, এবং পরের স্টেটমেন্টে প্রিন্ট করে দেখিয়েছে।

আউটপুট:

```
Name of the pet is: Thomas
```

আমরা যদি মূল প্রোগ্রাম [int main() রংগ] লক্ষ্য করি তাহলে দেখতে পাব, sayName এ আমরা cat অবজেক্ট পাস করে ফাংশনের মাধ্যমে cat এর নাম আউটপুটে দেখতে পাচ্ছি। যদি আমি cat.name কে cout করি তাহলে কিছুই দেখা যাচ্ছে না। কারণ কি? কারণ হল Pass by value।

ধরুন, আমি আপনাকে একটি ওয়েবপেইজের লিঙ্ক দিলাম (URL)। এখন যদি ওই লিঙ্কে কোন পরিবর্তন হয়, তাহলে পরিবর্তন যেমন আপনিও দেখতে পারবেন আমিও পারব। যদি আপনি আপনার পিসি থেকে লিঙ্কটি ডিলেট করে দেন তাহলে ওয়েবপেজের কিছুই হবে না। এটাই হল Pass by reference।

যদি আমি আপনাকে ওই URL এর একটি পেজ প্রিন্ট করে দেই তাহলে সেটি হবে Pass by value, যদি ওই URL এর ওয়েবপেজে কোন পরিবর্তন আসে তাহলে নিচয়ই আপনি আপনার প্রিন্টকৃত পেজটিতে দেখতে পাবেন না। এবং যদি ওটা আপনি নষ্ট করে দেন তাহলে আপনি নিজের কপিটি ধ্বংস করলেন। অর্জিনাল কপি ঠিকি তাদের ওয়েবসাইটে থাকবে।

উপরের প্রোগ্রামটি Pass by value হওয়ার কারণে মূল কপির কোন পরিবর্তন হয় নি।

একই প্রোগ্রাম আমি যদি `Pass by reference` দিয়ে লিখি তাহলে:

```
.....
void sayName(Pet &petObject, string petName) // I just put an & before the object!
{
    petObject.name = petName;
    cout << "Name of the pet is: " << petObject.name << endl;
}

int main()
{
    Pet cat;
    sayName(cat, "Thomas"); // This will print "Name of the pet is: Thomas"
    cout << cat.name << endl; // This will print "Thomas"
}
```

এটার আউটপুট:

```
Name of the pet is: Thomas
Thomas
```

`Pass by reference` এ আমি `cat` এর লোকেশন পাঠাই `&` চিহ্নটির মাধ্যমে। আমি `cat` অবজেক্টটি তৈরি করে তার লোকেশন ফাংশনে পাঠালাম এবং ফাংশনটি নতুন অবজেক্ট তৈরি না করে (আসলে নতুন অবজেক্ট তৈরি করলেও লোকেশন একই নির্দেশ করছে) `cat` অবজেক্টের লোকেশনের `name` টা পরিবর্তন করে দিল। `Pass by Value` তে যে অবজেক্টটি তৈরি করে অর্থাৎ, `petObject != cat [Not equal]` কিন্তু `Pass by reference` এ `petObject == cat` কারণ দুটি অবজেক্ট মূলত একটাই লোকেশন নির্দেশ করে আর সেটা হল `cat` এর লোকেশন। তাই `petObject` এর `name` পরিবর্তিত হওয়া মানে কিছুই না `cat` এর `name` পরিবর্তিত হওয়া যেহেতু তাদের লোকেশন একই!

কেস স্টাডি: [Led Blinking in Console using OOP!]

নিচের কোডটি রান করলে `Led blinking` দেখতে পাবেন কনসোলে, মানে কিছু বোরিং টেক্সট আপনাকে দেখাবে কত নাষ্ট পিনে `led` জুলছে, কত টাইম নিয়ে ব্লিংকিং করছে এই আরকি।

```

#include <iostream>
#include <windows.h>
using namespace std;

class Led
{
public:
    int ledPin;
    Led(int pinNumber);
    void pinMode(int);
    void blinkLed(int delay);
};

Led::Led(int pinNumber)
{
    ledPin = pinNumber;
    pinMode(pinNumber);
}

void Led::pinMode(int pin)
{
    cout << pin << " is set as output" << endl;
}

void Led::blinkLed(int delay)
{
    cout << "Led at " << ledPin << " is on" << endl;
    Sleep(delay);
    cout << "Led at " << ledPin << " is off" << endl;
    Sleep(delay);
}

int main()
{
    int usDelay, pinNumber, turn;
    cout << "Enter Delay, Pin number and Turns [sequentially, example: 1000 13 2]: ";
    cin >> usDelay, cin >> pinNumber, cin >> turn;
    Led led(pinNumber);
    while (turn > 0){
        led.blinkLed(usDelay);
        turn--;
    }
}

```

আউটপুট:

```
Enter Delay, Pin number and Turns [sequentially, example: 1000 13 2]: 1000 13 2
13 is set as output
Led at 13 is on
Led at 13 is off
Led at 13 is on
Led at 13 is off
```

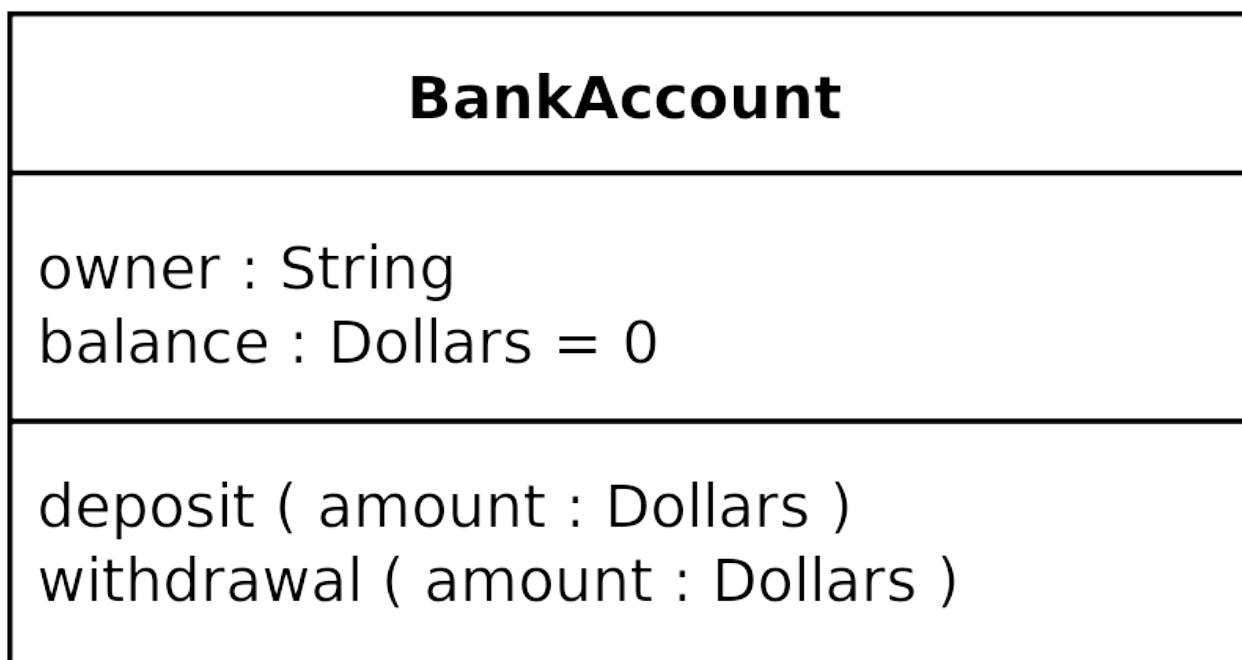
নোট:

Access Modifiers:

C++ এ তিনি ধরণের Access Modifiers আছে, public, private ও protected। এটা Encapsulation বা Data Hiding এর অঙ্গর্ত। তাই এখানে আলোচনা না করে পরবর্তী মূল পোস্টে আলোচনা করা হবে।

UML Diagram:

এখানে UML Diagram বলতে আমরা Class কে চিত্র দ্বারা প্রকাশ বুঝাব [Class Diagram]। এটি বিশাল জিনিস এবং এই সিরিজের আলোচ্য বিষয়বস্তু না। নিচে একটি ব্লাস ডায়গ্রাম ([উইকিপিডিয়া](#) থেকে সংগৃহীত):



- সবার উপরের অংশ হল Class Name। এটা বোল্ড হয় ও মাঝখানে থাকে, প্রথম বণ্টি Capitalized থাকে
- মাঝের অংশে Attributes বা Variable থাকে। Left aligned ও প্রথম বণ্ট Lowercase
- শেষের অংশে Method থাকে। Left aligned ও প্রথম বণ্ট
- Attributes ও Method এর আগে নিচের টেবিলের চিহ্ন থাকতে পারে [Access Modifier -> +, -, #;
Package -> ~, Derived-> /]

আজকে এই পর্যন্তই, পরবর্তী পরিচ্ছদে OOP এর বাকি বেসিক টপিকগুলো নিয়ে আলোচনা করা হবে।

অবজেক্ট ওরিয়েন্টেড প্রোগ্রামিং পরিচিতি (শেষ ভাগ)

গত পর্বে আমরা দেখেছিলাম ক্লাস ডিফাইন করে কিভাবে, অবজেক্ট তৈরি করে কিভাবে, অবজেক্ট এর মেথড, অ্যাট্ৰিবিউট সেট করা এবং তা `call` করা। আজকে আমরা বেশ কিছু জিনিস সংক্ষিপ্ত আকারে দেখব। আজকে যা আলোচনা করা হবে:

- Encapsulation
- Constructor
- Inheritance
- Polymorphism

Encapsulation:

এর অপর নাম `Data hiding` বা `Data protection`। OOP এর একটি জনপ্রিয় ফিচার এবং সফটওয়্যার ডিজাইন ও বিন্ডিংয়ের জন্যও গুরুত্বপূর্ণ। সফটওয়্যার ব্যবহারকারীর জ্ঞানার দ্রব্যকার নেই সফটওয়্যারটি কীভাবে কাজ করে (যারা নন ডেভলপার) কাজ চললেই এবং মনমত পার্ফর্মেন্স পেলেই হল। সেদিক থেকে `Encapsulation` খুবই জরুরি। কোন ক্লাসের Attribute গুলোকে সাধারণত আমরা `Encapsulated` করে থাকি। আমরা আগেই দেখেছি Object এর Attribute পরিবর্তন করা কোন ব্যাপার না। ধৰা যাক, Person একটি Class যার Attribute গুলো হল `name (string type)` এবং `age (int type)`। তাহলে আমরা Person এর একটা object তৈরি করে সহজেই তাৰ নাম পরিবর্তন কৰতে পাৰি:

```
#include <iostream>
using namespace std;

class Person
{
public:
    string name;
    int age;
};

int main()
{
    Person idiot;
    idiot.age = 0;
    idiot.name = "Idiot";
    cout << idiot.age << endl;
    cout << idiot.name << endl;
}
```

উপৰের কোডটি ঠিকঠাক চলবে কিন্তু নিচেরটা চলবেই না। পরিবর্তন কোথায়? `public` এর বদলে `private` বসিয়েছি :)

```

#include <iostream>
using namespace std;

class Person
{
private:
    string name;
    int age;
};

int main()
{
    Person idiot;
    idiot.age = 0;
    idiot.name = "Idiot";
    cout << idiot.age << endl;
    cout << idiot.name << endl;
}

```

তারমানে এভাবে আমরা Class কে Encapsulated করলাম! আব এতে সুবিধা হল ইউজার ডিরেষ্টলি তার Attribute এ অ্যাক্সেস পেল না।

এখনে আরেকটি গুরুত্বপূর্ণ ব্যাপার, যদি আমরা Attribute এ অ্যাক্সেস না পাই তাহলে সেটা দিয়ে আমাদের কি লাভ? এখনে আবার OOP সর্বোপরি C++ এর চমক। Private বা Protected ডেটা আমরা ডিরেষ্টলি অ্যাক্সেস করতে পারি না বটে, কিন্তু আমরা public ফাংশনের মাধ্যমে private ডেটায় অ্যাক্সেস পেতে পারি! তার মানে হল, আমার ক্লাসে যদি একটি পার্সিক মেম্বার ফাংশন থাকে এবং সেটা এমন একটি Attribute নিয়ে Deal করে যেটা কিনা Private/Protected তারপরও আমরা ওই Attribute এ অ্যাক্সেস পাব। নিচের উদাহরণটি দেখা যাক:

```

#include <iostream>
using namespace std;

class Person
{
public:
    void setNameAndAge(string n, int a);
    void showNameAndAge();
private:
    string name;
    int age;
};

int main()
{
    Person A;
    A.setNameAndAge("My Name", 20);
    A.showNameAndAge();
}

void Person::setNameAndAge(string n, int a)
{
    name = n;
    age = a;
}

void Person::showNameAndAge()
{
    cout << "The name: " << name << endl;
    cout << "The age: " << age << endl;
}

```

`setNameAndAge` এবং `showNameAndAge` দুইটা যেহেতু পারিক তাই আমরা `Object` দ্বারা ডিরেষ্টলি অ্যাক্সেস পাচ্ছি। আবার যেহেতু এই দুইটি মেথড ওই ক্লাসের মধ্যে অবস্থিত তাই ওই মেথডগুলো নিজের ক্লাসের সকল `Private/Public/Protected Attribute` গুলোতে অ্যাক্সেস পাবে। এখানে ব্যাপারটা ঠিক “কান টানলে মাথা আসে” মাথা প্রাইভেট তাই আমরা ডিরেষ্টলি মাথা টানতে পারব না, আমাদের হাতে যেহেতু কান আছে `(setNameAndAge এবং showNameAndAge)` তাই ওইটা দিয়েই টানতে পারব :D

এইখানে আরেকটি প্রশ্ন আসে, আমিতো নিজের ক্লাসের মেথড দিয়ে ডেটা পরিবর্তন করছি তাহলে `Encapsulation` এর দরকার কোথায়? `Encapsulation` দরকার কারণ `Public Data` পুরাই `Public` আর আমি অন্য কোন `Class` এর `Object` দ্বারা `Access` পেতে পারি।

```

#include <iostream>
using namespace std;

class Person
{
public:
    string name;
    int age;
};

class dataChanger
{
public:
    dataChanger(Person &p)
    {
        p.name = "I Changed it";
        p.age = 10;
    }
};

int main()
{
    Person person;
    person.name = "The Name has been set";
    person.age = 99;
    cout << person.name << endl;
    cout << person.age << endl;
    dataChanger danger(person);
    cout << person.name << endl;
    cout << person.age << endl;
}

```

আউটপুট:

```

The Name has been set
99
I Changed it
10

```

বিগিনারদের জন্য প্রোগ্রামটি একটু জটিল হতে পারে, এখানে দুইটি ক্লাস ব্যবহার করা হয়েছে, একটি হল `Person` এবং আরেকটি হল `dataChanger`। `dataChanger` এর মেথড একটাই এবং সেটা তার কনস্ট্রাইটর*। `dataChanger` এ আমরা একটি `Person Class` এর `Object` এর রেফারেন্স পাঠাই এবং তার `Attribute` গুলো পরিবর্তন করে দেয়। তাহলে দেখা যাচ্ছে `public` অ্যাক্টিভিটি গুলো আসলেই `Public` এবং তা অন্যান্য `ক্লাসের Method` দ্বারা `Accessible!`

আমরা যদি শুধু এই পরিবর্তনটা করি তাহলে অন্য ক্লাসের মেথড আর `Access` পাবে না:

```
class Person
{
private:
    string name;
    int age;
};
```

যেসব মেথড ও অ্যাক্টিভিটি Encapsulated করতে চান সেগুলোকে Access Modifier দ্বারা Modify করতে হবে। C++ এ কোন এক্সেস Modifier না ব্যবহার করলে ডিফল্ট Modifier হিসেবে private কাজ করবে। আপাতত Encapsulation নিয়ে এতটুকুই। এরপরে আমরা Constructor নিয়ে আলোচনা করব।

Constructor:

কোন ক্লাসের Constructor একটি ফাংশন ছাড়া কিছুই নয়। কিন্তু এর একটি বিশেষ আছে। Constructor এর প্রধান বিশেষ হল একে call করা লাগে না। Object তৈরি হওয়ার সাথে সাথে এই ফাংশনে যতগুলো স্টেটমেন্ট থাকবে সেগুলো অন্যান্য সাধারণ ফাংশনের মত Execute করতে থাকে এবং এটি একাধিক আর্গুমেন্ট নিতে পারে কিংবা নাও নিতে পারে, আরেকটি বৈশিষ্ট্য হল এটি কিছু Return করে না। তাহলে একনজরে Constructor এর বৈশিষ্ট্যগুলো হল:

- একে call করা লাগে না, অবজেক্ট তৈরি হওয়ার সাথে সাথেই বান হয়
- এক বা একাধিক Argument থাকতে পারে কিংবা আর্গুমেন্ট নাও থাকতে পারে
- কোন কিছু Return করে না
- Constructor যেহেতু ফাংশন তাই এই ফাংশনের নাম অবশ্যই সংশ্লিষ্ট ক্লাসের নাম হতে হবে

একটি উদাহরণ দেখা যাক:

```

#include <iostream>
using namespace std;

class Person
{
public:
    string name;
    int age;
    Person(string string_as_argument, int int_as_argument)
    {
        name = string_as_argument;
        age = int_as_argument;
    }
};

int main()
{
    Person A("Mr. A", 5);
    cout << "My name is: " << A.name << endl;
    cout << "My age: " << A.age << endl;
}

```

আউটপুট:

```

My name is: Mr. A
My age: 5

```

প্রোগ্রামটিতে কনস্ট্রাইটর কোনটি? আগেই বলেছি, কনস্ট্রাইটর হল সেই ফাংশন যার নাম আর ক্লাসের নাম একই এবং যেহেতু এটি কিছু রিটোর্ন করে না তাই তার আগে কোনপ্রকার Return Type এর কিওয়ার্ড থাকবে না। এখানে আমাদের ক্লাস হল Person এবং Person নামের একটি ফাংশন দেখা যাচ্ছে যার প্রথম আর্গুমেন্ট string টাইপ এবং দ্বিতীয় আর্গুমেন্ট int টাইপ। এখন চোখ বন্ধ করে বলে দেওয়া যায় কনস্ট্রাইটর আসলে কোনটি!

Person কনস্ট্রাইটরটি কি করছে আসলে? এটি দুইটি আর্গুমেন্ট নিয়ে অবজেক্ট এর ড্যারিয়েবলে তার মান বসাচ্ছে। অর্থাৎ আমরা যদি main ফাংশনের দিকে লক্ষ্য করি তাহলে দেখা যাবে আমরা Object তৈরি করার সময় দুইটা আর্গুমেন্ট পাস করেছি। একটি "Mr. A" এবং আরেকটি 5। Constructor আর্গুমেন্ট দুটো নিয়ে ক্লাসের name ও age ড্যারিয়েবলে বসিয়ে দিল। আদৌ বসালো কিনা তা দেখার জন্য আমরা A.name ও A.age দিয়ে অ্যাক্সেস করে আপনার কনস্ট্রাইটর কে চেক করলাম এবং আউটপুটে দেখলাম সেটা ঠিকঠাক আউটপুট দেখাচ্ছে।

আশা করি কিছুটা হলেও বুঝা গেল, Constructor যেহেতু ফাংশন তাই আমরা এটাকে নিচের মত করে বাইরে Declare করতে পারি!

```

class Person
{
public:
    string name;
    int age;
    Person(string, int);
};

Person::Person(string n, int a)
{
    name = n;
    age = a;
}

```

কনস্ট্রাক্টরের কিভাবে ব্যবহার করে তা নিয়ে দেখা গেল! এটি কতটা গুরুত্বপূর্ণ?

Constructor এর প্রয়োজনীয়তা:

কনস্ট্রাক্টরের মাধ্যমে আমরা অবজেক্ট এর Initial ভ্যালু বসিয়ে দিতে পারি, অন্য অর্থে আমরা যদি চাই যে একটি অবজেক্ট যখন তৈরি হবে তখন সেটা কিছু Initial Value নিয়ে তৈরি হোক। যাতে করে আমার সেই স্লাসের ড্যারিয়েবল ধরে ধরে ড্যালগুলো না বসানো লাগে। এবং সেই ড্যালগুলো ইনিশিয়ালি প্রসেসও করা যায় কনস্ট্রাক্টরের সাহায্যে।

আমরা দেখব Arduino তে LiquidCrystal লাইব্রেরি যখন আমরা ব্যবহার করব তখন কনস্ট্রাক্টরে কতগুলো পিন নাম্বার বসিয়ে দেব। ওই পিন নাম্বারগুলো অবশ্যই একটি নির্দিষ্ট Order এ দিতে হবে। সাধারণ Liquid Crystal Display চালানোর জন্য সাধারণত 14-16 পিন ব্যবহার করা হয়। এখানকার কিছু নির্দিষ্ট পিন আমরা Arduino এর সাথে কানেক্ট করি এবং ওই কানেক্ট করা পিনগুলো Order অনুসারে কনস্ট্রাক্টরে বসাই। বাকি সব কাজ হল LiquidCrystal লাইব্রেরির! অর্থাৎ ওই পিন অনুযায়ী LCD তে ReadWrite mode সেট করা, ক্যারেষ্টার শে করা ইত্যাদি সব সেটিংস কনস্ট্রাক্টরের মাধ্যমেই ঠিক হয়ে যায়। আমাদের নতুন কোন ফাংশন কল করা লাগে না কিংবা সেটা নিয়ে ঘাঁটাঘাঁটি করা লাগে না।

Inheritance:

ইনহেরিট্যান্স বলতে যা বুঝায় কাজে আসলেই তাই। আমরা প্রত্যেকেই আমাদের নিজ নিজ পিতামাতার কিছু বৈশিষ্ট্যধারণ করে থাকি। একই বংশের লোকজনদের মধ্যে সাধারণত কিছু Common বৈশিষ্ট্য থাকা অস্বাভাবিক কিছু নয়। দোষগুণ যা আছে তা বংশানুক্রমে চলতে থাকে। OOP এর আরেকটি চমৎকার ফিচার Inheritance! পুরনো কোড ব্যবহার করা এবং আপনার নতুন স্লাসকে আরও কিছু Method & Attribute যোগ করে আরও শক্তিশালী করে তোলা Inheritance এর অন্যতম কাজ।

আমরা Animal স্লাসে লক্ষ্য করি। Animal স্লাসের সদস্যরা কি করবে? খায়দায়, ঘুমায়, শিকার করে, চলাফেরা করে আরও অনেক কিছু করে। তাহলে আমরা সুবিধার্থে অল্ল কিছু ফাংশন ও অ্যাট্‌রিবিউট দিয়ে Animal Class টি ডিফাইন করি:

```
class Animal
{
public:
    void eat() { cout << "eating" << endl;}
    void sleep() {cout << "sleeping" << endl;}
    void hunt() {cout << "hunting" << endl;}
};
```

এখন যদি আমাকে বলা হয়, `Animal` ক্লাস বানাইলা ঠিকছে, এবার `Bird` নামের আরেকটি ক্লাস তৈরি কর।

`Bird` যেহেতু `Animal` এর মধ্যেই পড়ে এবং তার `eat`, `sleep` ও `hunt` এই মেথডগুলোও রয়েছে তাই আমি আগের ক্লাসের মেথডগুলো কপি করে পেস্ট করে দিতে পারি আর যেহেতু এসব কাজের পাশাপাশি `Bird` এর আরেকটি `Method` আছে আর সেটা হল `Fly` বা উড়া। :)

তাহলে ক্লাসটি দাঁড়াবে:

```
class Bird
{
public:
    void eat() {cout << "eating" << endl;}
    void sleep() {cout << "sleeping" << endl;}
    void hunt() {cout << "hunting" << endl;}
    void fly() {cout << "flying" << endl;}
};
```

কাজটা কি সুবিধাজনক হল? মোটেই না, যদি শখানেক মেথড ও অ্যাট্‌রিবিউট থাকে তাহলে কপি করতে করতেই অবস্থা খারাপ হয়ে যাবে! :P

তাহলে উপায় কী? `Inheritance Apply` করা :)

আমরা যদি এখন বলি `Bird inherits Animal` তাহলে `Animal` class র এর `eat`, `sleep` ও `hunt` মেথডগুলো অটোমেটিক চলে আসবে এবং তার জন্য আমার আর কিছুই করতে হবে না। যেটা অতিরিক্ত লাগবে সেটা শুধু অ্যাড করে দেব।

`Inheritance Apply` করার পরে কোডটি হবে:

```

#include <iostream>
using namespace std;

class Animal
{
public:
    void eat() { cout << "Eating" << endl;}
    void sleep() {cout << "Sleeping" << endl;}
    void hunt() {cout << "Hunting" << endl;}
};

class Bird : public Animal
{
public:
    void fly() { cout << "Flying" << endl; }

    // Calling the functions
    void activities()
    {
        eat();
        sleep();
        hunt();
        fly();
    }
};

int main()
{
    Bird b;           // Creating an object
    b.eat();          // Calling the functions individually
    b.sleep();
    b.hunt();
    b.fly();
    cout << "-----" << endl;
    b.activities();      // Calling the 'activities' function which calls o
}

```

আউটপুট:

```

Eating
Sleeping
Hunting
Flying
-----
Eating
Sleeping
Hunting
Flying

```

`Bird` এর ক্লাস থেকে আমরা দেখতে পাচ্ছি সেখানে `eat()`, `sleep()`, `hunt()` এর কোনটারই Definition দেওয়া নেই কিন্তু তারপরও ফাংশনগুলো আমরা `call` করতে পেরেছি। এই ফাংশনগুলো এসেছে `Animal` ক্লাসের থেকে। `Animal` ক্লাসে ফাংশনগুলোর পুরোপুরিভাবে ব্যাখ্যা করা হয়েছে তাই `Inherit` করার সময় আমাদেরকে ফাংশনগুলো নিয়ে মাথা ঘামাতে হয় নি, ডিরেচ্যুলি কল করেই কাজ করা গেছে।

ইনহেরিট্যান্স তাহলে প্রয়োগ করে কিভাবে?

```
class derivedClass : public baseClass { /* Put your attributes and methods here! */ };
// derivedClass is the inherited one
// class[keyword] your_class_Name :[colon] public/private/protected[access modifier] base
```

ইনহেরিট্যান্স ও বেশ বড় একটি টপিক, তাই অনেক কিছুই আলোচনা করা হল না। যতটুকু এখানে আলোচনা করা হল সেটুকুও এই কোর্সে লাগবে না আশাকরি।

Polymorphism:

`Poly` মানে বহু সেটা আমরা আগেই জানি। `Polymorphism` কে সেই হিসেবে বহুরূপতাও বলা যেতে পারে। একই জিনিসের বিভিন্ন রূপ থাকা মানেই সেই জিনিসটি বহুরূপ। `OOP` তেও বহুরূপ আছে:P

পলিমর্ফিজমও বিশাল একটি টপিক। আগেই বলে রাখা ভাল, `Function Overloading, operator overloading` কে সরাসরি `Polymorphism` বলা যাবে না। পলিমর্ফিজমের সংজ্ঞায় `Function/Method overloading` পড়ে না কিন্তু একে পলিমর্ফিজমের একটি অংশমাত্র হিসেবে বলতে ক্ষতি নেই। সবকিছু সিম্পল রাখার জন্য আমি এখানে `Function Overloading` টা দেখাব এবং আপাতত একেই `Polymorphism` হিসেবে বলব। তারপরও আবারও দ্রষ্টব্য, `Function overloading` মানেই কিন্তু `Polymorphism` নয়।

আলোচনা শুরুর আগে নিচের প্রোগ্রামটি দেখা যাক:

```

#include<iostream>

using namespace std;

class Shape
{
public:
    float calcArea(int height, int width); // Calculate area for Rectangular shape
    float calcArea(double base, double height); // Calculate area for Triangular Shape
    float calcArea(int side); // Calculate area for Square shape
};

float Shape::calcArea(int height, int width)
{
    return height * width;
}

float Shape::calcArea(double base, double height)
{
    return 0.5 * base * height;
}

float Shape::calcArea(int side)
{
    return side * side;
}

int main()
{
    Shape rectangle, triangle, square;
    cout << "Area of rectangle is: " << rectangle.calcArea(5,3) << endl;
    cout << "Area of triangle is: " << rectangle.calcArea(2.0,5.0) << endl;
    cout << "Area of square is: " << square.calcArea(5) << endl;
}

```

আউটপুট:

```

Area of rectangle is: 15
Area of triangle is: 5
Area of square is: 25

```

দেখা যাচ্ছে ক্লাস `Shape` এ আমরা তিনটা মেথড তৈরি করেছি একই নামের `calcArea` যার কাজ হল ক্ষেত্রফল বের করা। ভাল করে লক্ষ্য করলে দেখা যাবে তিনটি ফাংশনের নাম এক হতে পারে কিন্তু প্রত্যেকটার সাথে প্রত্যেকটার পার্থক্য বিদ্যমান। পার্থক্যগুলো হল, প্রথম ফাংশনটির আর্গুমেন্ট ২ টা এবং ২টাই `int` টাইপের। পরের ফাংশনটিরও আর্গুমেন্ট ২টা কিন্তু ২টার টাইপ হল `double` এবং শেষের ফাংশনটির আর্গুমেন্ট কেবল ১টি।

নাম এক হলেও আর্গুমেন্টের ভিত্তিতে ফাংশনগুলো আলাদা। কিন্তু কম্পাইলার কিভাবে বুঝবে আমি আসলে কোন ফাংশনটা `call` করছি? `c++` কম্পাইলার এতটাই স্মার্ট যে সে শুধু আপনার দেওয়া আর্গুমেন্ট টাইপ বা কয়টা আর্গুমেন্ট দিয়েছেন সেটা দেখেই বুঝতে পারবে আসলে আপনি কোন ফাংশনটি `call` করছেন। এই পদ্ধতির কেতাবি নাম `Function / Method overloading`। আমরা আপাতত একেই পলিমর্ফিজম হিসেবে চিহ্নিত করছি।

আর এখানেই অবজেক্ট ওরিয়েন্টেড প্রোগ্রামিংয়ে ইতি টানলাম। OOP এর অনেক ফিচার বাদ পড়েছে ও জিনিসগুলো সাধারণ রাখার জন্য অনেক ড্রলডাল কথা লিখেছি। আড্রিনো প্রোগ্রামিংয়ে OOP এর ব্যবহার যদিও আমরা কদাচিৎ দেখব তারপরও এই বিষয়গুলোতে ধারণা রাখা জরুরি। পরের পরিচ্ছদে আমরা সিরিয়াল কম্যুনিকেশন সম্পর্কে মোটামুটি জানার চেষ্টা করব। সেখানে অবজেক্ট ওরিয়েন্টেড প্রোগ্রামিংয়ের সব ব্যাপার না থাকলেও সাধারণ ধারণা থাকলে বুঝতে অসুবিধা হবে না। তাছাড়া পরবর্তীতে `Header` ফাইল তৈরি করা ও এর ব্যবহারও দেখানো হবে, তখন OOP জানলে কোড অনেকাংশে সংক্ষিপ্ত ও `Readable` ও `organized` করা সম্ভব।

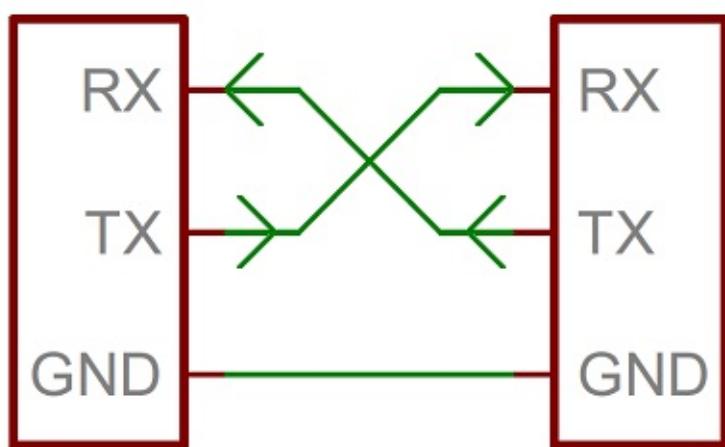
সিরিয়াল কম্যুনিকেশন - সংক্ষিপ্ত সংস্করণ

গত দুই পর্বে আমরা অবজেক্ট ওভিয়েটেড প্রোগ্রামিং সম্পর্কে হালকা পাতলা ধারণা লাভ করেছিলাম। আজকের বিষয়বস্তু হল সিরিয়াল কম্যুনিকেশন।

যা যা লাগবে:

- Arduino Board (Uno / Mega 2560)
- Arduino IDE (-_-)
- Usb Cable
- Computer (duh!)

সিরিয়াল কম্যুনিকেশন (Serial Communication):



সিরিয়াল কম্যুনিকেশন হল এমন একটি ব্যবস্থা যার মাধ্যমে আপনি আপনার প্রিয় বোর্ডের সাথে কথা বলতে পারেন। লিটারেলি কথা? না ঠিক তা নয়, আপনি কিছু ডেটা পাঠাবেন সে সেটাকে প্রসেস করবে তারপর সেও কিছু তথ্য পাঠাবেন সেটা আবার আপনি দেখতে পারবেন কিংবা প্রসেস করতে পারবেন। অনেকটা কথা বলার মতই শুধু ডাষ্টাটা হল ০ আর ১। আর এই কথা বলার জন্য আপনার অবশ্যই আরেকটা ডিভাইস দরকার যার সাথে আডুইনো কানেক্টেড থাকতে পারবে এবং তার পাঠানো ডেটা সে বুঝবে।

সিরিয়াল কম্যুনিকেশন একটা মাধ্যম মাত্র। আরও অনেক ধরণের কম্যুনিকেশন সিস্টেম আছে, আডুইনোতেই; কিন্তু সেগুলো আলোচনা করার সময় এখনো আসে নি। এই কোর্সে যতটুকু দরকার সেটুকুই শুধু “সংক্ষিপ্ত সংস্করণ” – এ থাকবে, যাতে জিনিসটা একঘেয়ে না হয় ও যতটা আলোচনা করা হয় ততটুকুই ডালভাবে শিখতে পারেন।

আমরা যখন আডুইনো বোর্ড কনফিগার করছিলাম তখন দেখেছি সেটা একটা COM Port এ সংযুক্ত থাকে, যেমন COM3, COM14 ইত্যাদি। এই পোর্টকে আমরা দরজাও বলতে পারি। যে দরজা দিয়ে ডেটা যাওয়া আসা করে। আমরা যে কোড আপ্লোড করি, চিন্তা করুন তো, কীভাবে আপ্লোড হ্য? হ্যাঁ, সিরিয়াল কম্যুনিকেশনের মাধ্যমে, কম্পাইল্ড

.hex ফাইলটা বিভিন্ন প্রসেসের মধ্য দিয়ে চিপে প্রবেশ করে। আর্ডুইনোর ক্ষেত্রে সিরিয়াল কম্যুনিকেশন এ সাহায্য করার জন্য ATmega16u2 কিংবা ATmega8u2 চিপটি কাজ করে। এখানে আরও অনেক জটিল ব্যাপার স্যাপার আছে।

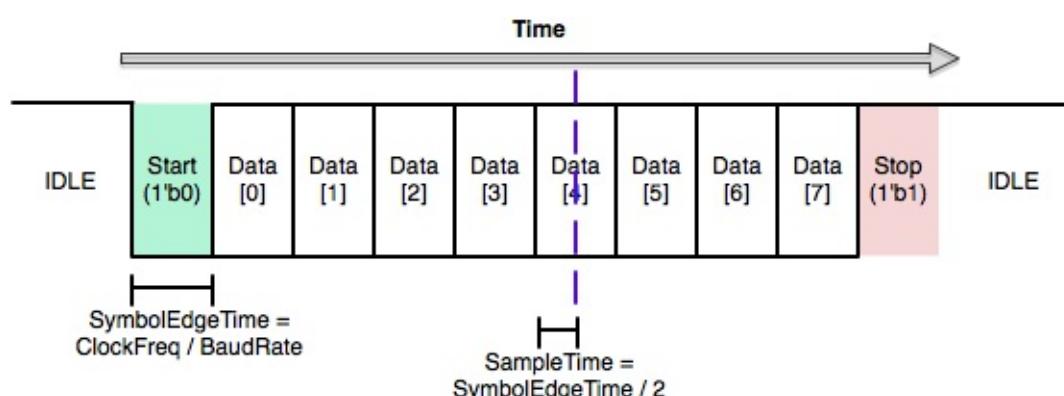
তাহলে বুঝলাম ডেটা পাঠানোর একটা টার্ম হল Serial ! কিন্তু ডেটা ট্রান্সফার কীভাবে হচ্ছে?

এই প্রশ্নের উত্তর দিতে গেলে আমাদের আরেকটু জ্ঞানলাভ করতে হবে। Hardware এর ক্ষেত্রে এই ডেটা ট্রান্সফার করার জন্য বেশকয়েকটি প্রোটোকল আছে, এই প্রোটোকল গুলোর মধ্যে সাধারণত সর্বাধিক ব্যবহৃত প্রোটোকলটি হল UART (Universal Asynchronous Receiver Transmitter)। এই পদ্ধতিটা অনেক পুরাতন এবং পুরাতন বা নতুন সব চিপেই ব্যবহারযোগ্য।

UART তে 8-bit ডেটা ট্রান্সফার করা হয়, একটা থাকে লো লেভেল স্টার্ট বিট এবং 8 টা ডেটা বিটস ও আরেকটি হাই লেভেল স্টপ বিট। সিরিয়াল কম্যুনিকেশন চালু করার জন্য লো লেভেল স্টার্ট বিট ও হাই লেভেল স্টপ বিট ব্যবহার করা হয়।

UART এর আরেকটি সুবিধা, কোন নির্দিষ্ট ডোল্টেজ লেভেলে ডেটা ট্রান্সফার করতে হবে এমন কোন বাধ্যবাধকতা নেই। 3.3V কিংবা 5V যে লজিকেই আপনার MCU চলুক না কেন তাতেই হবে। এখানে গুরুত্বপূর্ণ একটি কথা বলার দরকার, আপনি যে ডিডাইসের সাথে যোগাযোগ করতে চাচ্ছেন এই প্রোটোকলের মাধ্যমে, দুই ক্ষেত্রে Transmission speed * সমান হতে হবে।

UART এর মাধ্যমে ডেটা পাঠানো হয় যেভাবে:



ধরা যাক, UART Protocol এর মাধ্যমে আমি “hello” ওয়ার্ড টি Boka নামক MCU তে পাঠাতে চাচ্ছি। তাহলে আমি যখন পাঠাব তখন এর সাথে একটি Start Bit বা ট্যাগ লাগানো হবে যাতে অন্য ডিডাইসটি বুঝতে পারে ডেটা ট্রান্সফার শুরু হতে যাচ্ছে এবং ডেটা রিসিভ করার জন্য তার তৈরি হতে হবে। কথা হল ডেটা পাঠাবো আমি পিসি থেকে, তার কান খোলা রাখলেই তো হয়, তার জন্য তাকে আবার তৈরি হতে হবে কেন?

বাস্তবে আমরা অনেক সময় অনেকের কথা ঠিকমত বুঝি না, মোটমাট এর কারণ হতে পারে পাঁচটি

- হয় সে তাড়াতাড়ি কথা বলছে
- অথবা এত ধীরে কথা বলছে যে শোনার আগ্রহ হারিয়ে ফেলি তাই বুঝি না
- আমি বুঝিনা এমন ডাষায় কথা বলছে
- তার কথার কোন মানে নেই
- আরেকজন যখন কথা বলছে তখন আমি শুনছি না কিংবা আমি যখন কথা বলছি সে শুনছে না

UART/USART এর ক্ষেত্রে কম্যুনিকেশনে যে ঝামেলাটা হয় সেটা হল প্রথম দুইটা এবং শেষেরটা। Serial Communication Establishment এর জন্য একজনের বলার গতি ও আরেকজনের শোনার গতি সমান হতে হবে। যদি তা না হয়, তাহলেই PC একটা কথা বললে Boka বুঝবে উচ্চটা। কিংবা PC যখন কথা বলছে তখন Boka যদি কান খাড়া না করে বা vice versa না হলে ডেটা পাঠানো সম্ভব হবে না। এইকারণে PC যখন ডেটা পাঠাচ্ছে তখন Boka কে রিক্যুয়েস্ট করছে তার ঘড়ি (clock) টা যেন PC এর সাথে মিলিয়ে synchronized করে নেয়। ধরা যাক Boka MCU তার Clock ঠিকঠাক করে নিল এবং সে “hello” ডেটাটি নেওয়ার জন্য প্রস্তুত। আমরা যেহেতু প্রোগ্রামিংয়ে এক্সপার্ট BI তাই আমরা অনেক আগে থেকেই জানি যে String হল char টাইপের Null Terminated Array মাত্র। তাই String পাস করতে হলে একটা একটা করে ক্যারেক্টার পাঠাতে হবে আগে, তাই না?

এইবার ডেটা ট্রান্সফারের সময় যদি ঘড়ির ব্যাটারি চলে যায় এবং দুইটা ঘড়ি synchronized না হয় তবে আবারও ডেটা ট্রান্সফারে সমস্যা হবে। পরীক্ষা করে(!) দেখা গেছে PC ও Boka MCU এর মধ্যে সুষ্ঠু ডেটা ট্রান্সফার করার জন্য তাদের ঘড়িয়ে শতকরা ৮০ ভাগ Accurate হতে হবে।

আবারও মূল কথা থেকে সরে আসার জন্য দৃঃখিত। >:(যাকগে যেটা বলছিলাম আরকি, Start bit দিল এবং বলল আমি ডেটা পাঠাচ্ছি, এইবার Boka MCU Clock ঠিক করা মাত্র PC পাঠানো শুরু করল। এই ডেটাণ্ডলোই হল সেই কাঙ্ক্ষিত ডেটা যেটা আমরা পাঠাতে চাচ্ছি। এণ্ডলোকে আমরা আঁতলামি করলে বলো LSB (Least Significant Bit)। মাথায় রাখতে হবে প্রতিটা বিট পাঠানোর সময় পুরো এক, মানে যদি একটা বিট পাঠাতে ২ মাইক্রোসেকেন্ড লাগে তাহলে তার পরের বিট পাঠাতেও একই সময় লাগবে। রিসিভার বা Boka MCU একটা নির্দিষ্ট টাইম পরপর চেক করে দেখবে কি বিট আসছে এবং এই সময়টা হবে PC এর বিট পাঠানোর পর্যায়কালের অর্ধেক সময় বা এই উদাহরণে ১ মাইক্রোসেকেন্ড।

এখানে আবার একটা কথা আছে, PC তো আর জানে না Boka MCU কখন চেক করে, PC শুধু জানে কখন তার ট্রান্সমিশন শুরু করতে হবে এবং কতক্ষণ পরপর বিট পাঠাতে হবে।

বিট পাঠাতে পাঠাতে যখন পুরো “hello” টাই PC পাঠিয়ে দিল সে এবার “hello” এর লেজের সাথে একটা Parity Bit পাঠিয়ে দিতে পারে [may but not will]। এই Parity Bit টা কিন্তু Stop Bit না, এটা হল Error checking Bit। এই বিট Receiver [Boka MCU] Error চেকিংয়ের কাজে ব্যবহার করতে পারে। Parity Bit পাঠানোর পরপরই সে একটা Stop Bit পাঠিয়ে দেবে।

এই Parity Bit টা জেনারেট করার দায়িত্ব Transmitter এর কিন্তু Parity Bit টা কি হবে, ব্যবহার করা হবে কি হবে না সেটা নির্ভর করে PC ও Boka MCU এর সময়োত্তর উপর। এখন ধরি সম্পূর্ণ word পাঠানোর পর যথাসময়ে Boka MCU Stop Bit টি পায় নি। তাহলে Boka MCU বোকার মত (নাকি চালাকের মত?) যতটুকু ডেটা পেয়েছে সম্পূর্ণটাই গার্বেজ ড্যালু হিসেবে ধরবে এবং Framing Error * দেখাবে।

তাই UART Protocol এর জন্য সবকিছু কনফিগার করে নেওয়াই ভাল। কারণ UART অটোমেটিক্যালি সবকিছু সেট করে নিতে পারে না, আর যদি Sender ও Receiver কে Identically configure না করা হয় তাহলে Start, Stop কিংবা Parity Bit এগুলো হোল্টে পাস হ্যান্ড না।

আবারও ধরি “hello” ভালভাবেই গেল, কিন্তু আমি এখন “world” পাঠাব, তাহলে কী করব? কিছু করা লাগবে না, কনফিগারেশন ঠিক থাকলে খালি পাঠালেই হবে, PC আবারও একটা Start Bit পাঠিয়ে বলবে আরেকটি ডেটা আসছে বেড়ি হও এবং Stop Bit পাঠিয়ে বলবে ডেটা পাঠানো শেষ।

তাই যখন ডেটা পাঠানো হয় না তখন `Transmission line` কে জন্ম নাইন বলতে আপত্তি নেই। `USART` প্রায় একই ভাবে কাজ করে কিন্তু এর কাজের ধরণ একটু আলাদা এবং উন্নত। `USART` নিয়ে আরেকটি পরিচ্ছদে কথা বলা যাবে।

যাই হোক সিরিয়াল কম্যুনিকেশন নিয়ে আমরা অগাধ জ্ঞানলাভ করলাম, চলুন দেখি এটাকে এবার অ্যাপ্লাই করতে পারি কিনা।

আর্ডুইনোতে সিরিয়াল কম্যুনিকেশন:

আর্ডুইনোতে সিরিয়াল কম্যুনিকেশন তৈরি করা অনেক সহজ, এর কারণ অনেকগুলো, তার মধ্যে দুইটা কারণ হল `সিরিয়াল কম্যুনিকেশনের জন্য ডেভিকেটেড ATmega8u2 বা ATmega16u2 চিপ এবং Arduino.h ফাইলটা।`

আমরা শুধু দেখব সিরিয়াল কম্যুনিকেশনের মাধ্যমে কীভাবে পিসির সাথে আর্ডুইনো কথা বলতে পারে। তাহলে ঝটপট নিচের কোডটি কপি পেস্ট করে `Arduino IDE` তে বসিয়ে দিন। আমিও জানি আপনিও জানেন যে কোডটা নিজে লেখার মত সময় আপনার নাই, সময় না থাকলে কি করা; কপি পেস্ট করেন, আপনার ক্ষতি হইলে আমার কি :P

কোডটা আশোড করন এবং আর্ডুইনো বোর্ড কানেক্টেড রাখুন!

```
const int led = 13;
char command;

void setup() {
  Serial.begin(9600);
  Serial.println("Ready for command\nEnter A/a to turn on\nEnter other keys to turn off");
  pinMode(led, OUTPUT);
}

void ledOn()
{
  Serial.println("Led On!");
  digitalWrite(led, HIGH);
}

void ledOff()
{
  Serial.println("Led Off!");
  digitalWrite(led, LOW);
}

void loop() {
  if (Serial.available()){
    command = Serial.read();
    if (command == 'a' || command == 'A') ledOn();
    else ledOff();
  }
}
```

আউটপুট:

```

File Edit Sketch Tools Help
sketch_feb22a §
pinMode(led, OUTPUT);
}

void ledOn()
{
    Serial.println("Led On!");
    digitalWrite(led, HIGH);
}

void ledOff()
{
    Serial.println("Led Off!");
    digitalWrite(led, LOW);
}

void loop() {
}

Done uploading.
Sketch uses 2,370 bytes (7%) of program storage space.
Maximum is 32,256 bytes.

Global variables use 269 bytes (13%) of dynamic memory,
leaving 1,779 bytes for local variables. Maximum is 2,048

```

আউটপুট আরকি, নিজেই দেখে নিন।

প্রোগ্রাম Walkthru:

যেগুলোর ব্যাখ্যা দেওয়া নাই মনে হচ্ছে, পূর্বের পোস্টগুলোতে দেওয়া হয়েছে।

Line 5:

```
Serial.begin(9600);
```

OOP এর সূত্রমতে পাই, Serial হল একটি অবজেক্ট যার Definition "Arduino.h" হেডার ফাইলটিতে দেওয়া আছে। begin হল Serial এর একটি মেথড। OOP এর সময় বলেছিলাম, কোন অবজেক্টের মেথড ডাকার জন্য objectName.methodName(parameter) এই রূল ফলো করতে হয়। এখানেই ঠিক তাই করা হয়েছে, begin এর কাজ হল Serial communication establish করা।

begin মেথডে আমরা একটি ইন্টিজার ড্যালু দেখতে পাচ্ছি 9600 ! এটি হল Baud Rate। মানে কত স্পিডে আড়ুইনো পিসির সাথে কথা বলবে? UART এর ক্ষেত্রে আমরা জানি স্পিড সমান না হলে ঝামেলা তাই আমরা আড়ুইনো বোর্ডের স্পিড ঠিক করে দিলাম। আর পিসির স্পিড ঠিক করলাম Serial Monitor এর মাধ্যমে ;)

Line 6:

```
Serial.println("Ready for command\nEnter A/a to turn on\nEnter other keys to turn off")
```

println -> এই ফাংশনের কাজ হল সিরিয়াল মনিটরে কিছু প্রিন্ট করা। অর্থাৎ কিনা printf("Hello world") এর আড়ুইনো Serial comm ভার্সন B। print কমাণ্ড ও একই কাজ করবে তবে println ফাংশনটি কিছু প্রিন্ট করার পাশাপাশি একটা Enter বসিয়ে দেবে বা নিউলাইন প্রিন্ট করবে। যাতে আমাদের পড়তে সুবিধা হয়।

অর্থাৎ println("blablabla") হল print("blablabla\n") এর সমতুল্য।

Line 11:

```
void ledOn()
{
    Serial.println("Led On!");
    digitalWrite(led, HIGH);
}
```

এই ফাংশনটির কাজ হল সিরিয়ালে সে প্রিন্ট করবে “বাতি জুলাইসি” ও বাতি জুলিয়ে দেবে। পরের ফাংশনটি কি করবে সেটা অনুমান করুন (বাড়ির কাজ) :P

Line 24:

```
if (Serial.available()) {
```

এই available মেথডের কাজ হল চেক করে দেখা, আড়ুইনো কি ডেটা গ্রহণে তৈরি কিনা? যদি হ্য তাহলে তাকে পাঠাব নইলে নয়। আরও একটি মানে আমরা বের করতে পারি, available() মেথডের রিটার্ন টাইপ boolean বা bool।

Line 25:

```
command = Serial.read();
```

অর্থাৎ, সিরিয়ালে যদি আমি কিছু ইনপুট দেই তা আডুইনো পড়ে `command` নামক ভ্যারিয়েবলে রাখবে। তারমানে `read()` মেথডের রিটার্ন টাইপ `char` (আপাতত এটাই জেনে রাখুন)

Line 26 ও Line 27:

```
if (command == 'a' || command == 'A') ledOn();
else ledOff();
```

যদি সিরিয়ালে যেটা পাইসি সেটা `a` বা `A` এর সমান হয় তাহলে বাতি জুলাও। নইলে বাতি নিভাও :D

একনজরে পুরো প্রোগ্রাম:

- একটা ফ্রব সংখ্যা ১৩ নিলাম যার নাম `led`
- একটা ক্যারেক্টার টাইপ ভ্যারিয়েবল নিলাম যার নাম `command`
- `Serial communication` চালু করলাম `9600 bits per sec` স্পিডে
- `led` কে আউটপুট বানালাম
- একটা ফাংশন তৈরি করলাম `ledOn` নামে যার কাজ বাতি জুলানো ও আমাকে জানানো বাতি জুলেছে কিনা
- আরেকটা ফাংশন তৈরি করলাম `ledOff` নামে যার কাজ বাতি নিভানো ও আমাকে জানানো বাতি নিডেছে কিনা
- ইনফিনিটি লুপে প্রবেশ:
- যদি সিরিয়াল পাই, তাহলে সিরিয়ালে যা ইনপুট দেয় তা `command` এ রাখলাম
- `command` যদি `a` বা `A` এর সমান হয় তাহলে `ledOn()` ফাংশনৰে ডাক দে
- যদি তা না হয় তাহলে `ledOff()` ফাংশনৰে ডাক দে

সিরিয়াল নিয়ে ব্যাপক জ্ঞান দিলাম, পরবর্তীতে `ADC` এর সময় কাজে দেবে। তখন আমরা আরও নতুন কিছু দেখব।

নোট:

Baud Rate:

এর মানে কত বিট হাবে ডেটা পাস হচ্ছে। `Arduino` বা সাধারণত ডিভাইসগুলোর `Baud Rate` হয় `9600 bits per second`। আডুইনোর সর্বোচ্চ `Baud Rate 115200 bits per second`।

Framing Error:

যদি `PC` ও `MCU` তে সিঙ্গেলাইজেশন জাতীয় সমস্যা দেখা যায় কিংবা ডেটা সিগনাল `Interrupted` হয় তখন এই সমস্যাটা দেখায়।

সচরাচর জিজ্ঞাস্য প্রশ্ন [F. A. Q]

Serial Monitor কী জিনিস? কী কাজে লাগে?

ইয়ে Serial দেখার কাজে লাগে B। সিরিয়াল মনিটর অনেক কাজে লাগে, সিরিয়ালে ডেটা পাঠানো, সিরিয়ালে নামক ওয়ালে ডিভাইসটা কি স্ট্যাটাস দিল তা দেখা যায়। কমান্ড পাঠানো যায়। ইত্যাদি কাজ করার জন্য Serial Monitor লাগে, আডুইনোর বিল্ট-ইন সিরিয়াল মনিটর ছাড়াও অনেক ফ্রি ও ওপেনসোর্স সিরিয়াল মনিটর আছে বহুত ফাংশনসহ। আপাতত আমদের তেমন ফিচার লাগে নাই বলে আমরা আডুইনোর বিল্ট-ইন সিরিয়াল মনিটর ব্যবহার করছি।

Serial Communication এর উপকারিতা কী?

উপকারিতার কথা বলতে পারব না কিন্তু অপকারিতা নাই সেটা জানি :P। Sensor checking, wireless/wired data transmission, interfacing কিসে লাগে না সেটা বলেন! আস্তে আস্তে উপকারিতা বুঝতে পারবেন আশা রাখি।

টাইটেলটা বুঝলাম না “সিরিয়াল কম্যুনিকেশন (সংক্ষিপ্ত সংস্করণ)” – মানে??

মানে হল এখানে যা আলোচনা করা হয়েছে তা Serial communication এর ধারেকাছেও নাই, এর যে 8-5 টা প্রোটোকল আছে তার প্রত্যেকটা নিয়ে বিশাল বিশাল বই লেখা সম্ভব। বাকি প্রোটোকলগুলো আপাতত লাগছে না বলেই সেগুলো এখান থেকে বাদ দেওয়া হয়েছে। আরেকটা সংস্করণ করে সেখানে বাকি প্রোটোকল নিয়ে হালকা আলোচনা করার ইচ্ছা আছে।

আপাতত এই পর্যন্তই, পরের পরিচ্ছদে আমরা ADC শিখব!

আডুইনোতে অ্যানালগ থেকে ডিজিটাল কনভার্সন

গত আমরা মোটামুটিভাবে বেসিক জিনিসগুলো প্রায় শেষ করে ফেলেছি। সব বেসিকগুলোর মধ্যে, আজকেরটা অনেক গুরুত্বপূর্ণ। লাইন ফলোয়ার, মেজ সল্ডার ইত্যাদি বোবট বানাতে তো বটেই, Environment Interaction বা পরিবেশের পরিস্থিতির তারতম্যর সাথে যেকোন কাজ করার জন্য ADC মাস্ট। তাহলে দেরি না করে আজকের জন্য দরকারি জিনিসপত্র যোগাড় করে ফেলুন। যা যা লাগবে:

- 10K / 20K Potentiometer (AKA Variable Resistor)



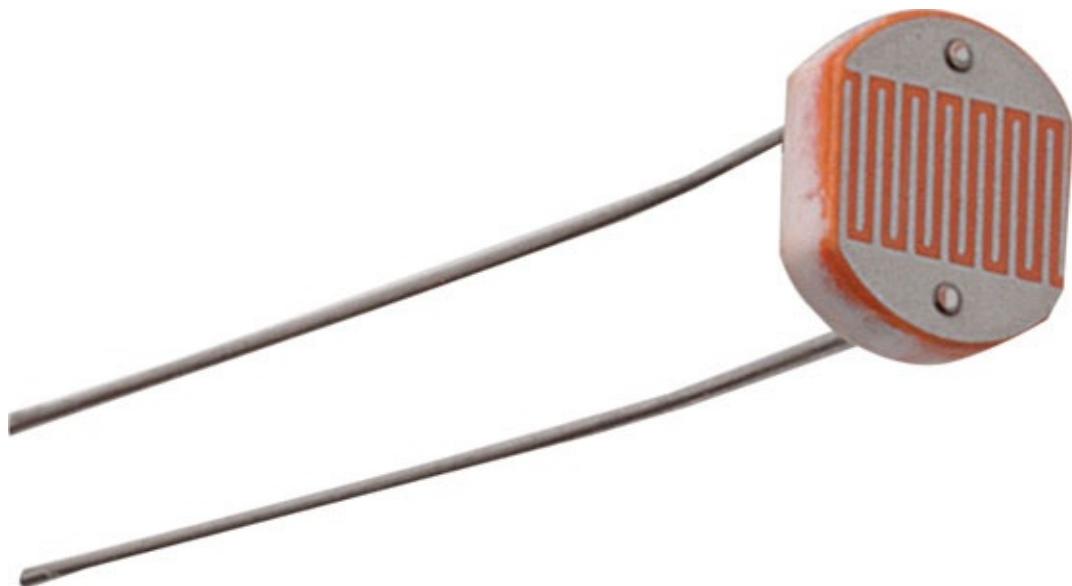
- Premium Jumper Wires [22AWG Solid would work too]



- IR Sensor pair [RX/TX] [যেকোন অ্যানালগ IR হলেই হবে, সাধারণত যেসকল জোড়ায় ২-২ পা থাকে সেগুলোই অ্যানালগ IR এবং এরা অ্যানালগ ড্যালু দিয়ে থাকে। যদি আপনার IR এর যেকোন একটির দুইটির বেশি লেগ বা পা থাকে তাহলে বুঝতে হবে আপনার IR টি অ্যানালগ না, ডিজিটাল। ৩ লেগবিশিষ্ট IR পাওয়া যায় চিনিতে। ডিজিটাল IR থেকে শুধু ০ বা ১ আউটপুট পাওয়া যায়।]



- IR Alternative LDR-LED Pair [Light Dependent Resistor]



- Resistors [1k, 330, 10k]

ADC শুরুর আগে:

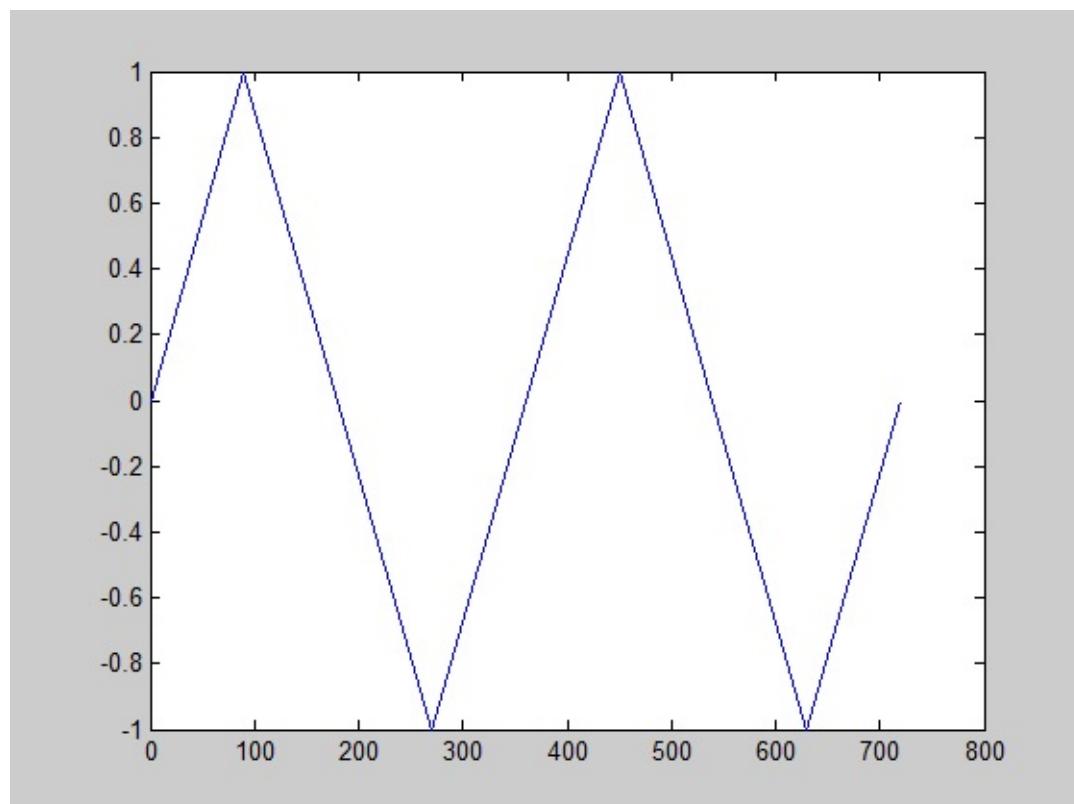
অ্যানালগ কি জিনিস আৰ ডিজিটাল কি জিনিস? তাৰ আগে আৱেকটি ব্যাপৰ সম্পর্কে ধাৰণা নেয়া যাক।
মাইক্ৰোপ্ৰসেসৰ কিংবা মাইক্ৰোট্ৰোলৱ আপনাৰ মত মোটেই বুদ্ধিমান না। সে শুধু বুঝতে পাৰে ভোল্টেজ আছে কি নাই। ০ নাকি ১? শুধু এটাই সে বুঝতে পাৰে। ০ এবং ১ ছাড়াও অসংখ্য State আমৰা প্ৰকৃতিতে তাকালৈ দেখতে পাই। প্ৰকৃতিৰ দৰকাৰ কী? একটু ভাবলৈ তো হয়। যেমন, ১ থেকে ১০ পৰ্যন্ত গণনা কৰাৰ কথাই চিন্তা কৰা যাক। এভাবে আমৰা গুণতে পাৰি, ১, ২, ৩ ... অথবা ১, ৩, ৫, ৭ ... অথবা ১, ১.১, ১.২, ১.৩, ১.৪ এইভাবেও গণনা কৰতে পাৰি।

হ্যত ভাবছেন এখানে লক্ষ্য কৰাৰ কী আছে? অবশ্যই আছে, ১ থেকে ১০ এৰ দূৰস্থ নিৰ্দিষ্ট! কিন্তু এটাকে বিভিন্ন ভাবে আমৰা মাপতে পাৰি। সেটা হল, ১ থেকে ১০ এৰ মধ্যকাৰ দূৰস্থকে কতগুলো ভাগ কৰে। আচ্ছা এবাৰ ভাবুন তো, গণনাৰ ক্ষেত্ৰে কোনটি বেশি এফিষিয়েন্ট? ১, ২, ৩ নাকি ১.১, ১.২, ১.৩? আসলে জিনিসটা নিৰ্ভৰ কৰে আমি আসলে আমাৰ গন্তব্যস্থলে যাওয়াৰ জন্য কতগুলো স্টেপস চাচ্ছি? একটি কৰে স্টেপস নিয়েও সিডি দিয়ে উঠা যায় আবাৰ ডাবল স্টেপ একবাৰে নিয়েও উঠা যায়। আমাদেৰ যখন যেমন দৰকাৰ আমৰা তখন তেমন স্টেপস নিয়ে থাকি।

আমৰা এখন ADC এৰ সাথে এই গণনা জিনিসটাৰ ভাগেৰ সাথে একটা সম্পৰ্ক তৈৰি কৰব। কিন্তু তাৰ আগে একটু আঁতলামি কৰা প্ৰয়োজন।

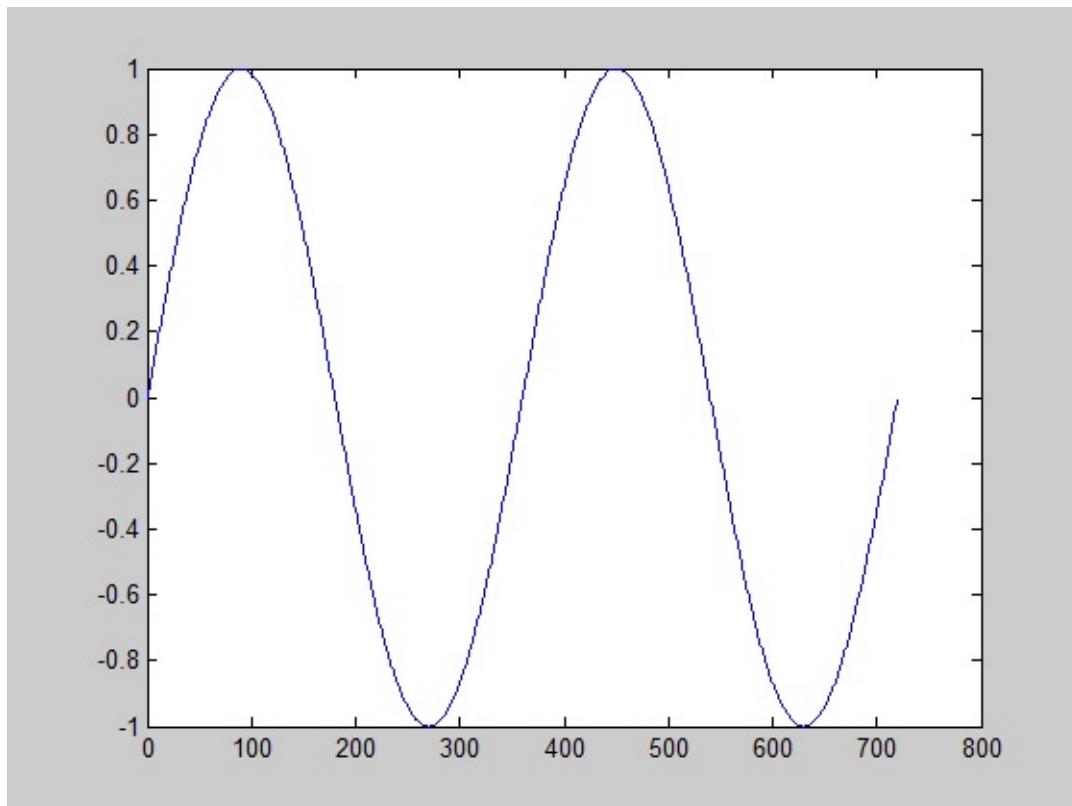
ধৰন, আপনাকে `sin(theta)` এৰ কাৰ্ড আঁকতে দেওয়া হল `vs theta`। মানে কোণেৰ সাথে `sin(কোণ)` এৰ সম্পৰ্ক দেখানোৰ জন্য। আৱও ধৰন, আপনাকে প্ৰতি ৯০ ডিগ্ৰি গ্যাপে গ্যাপে বিন্দু বসাতে হবে। একবাৰ ০ ডিগ্ৰিৰ বিপক্ষে বসালেন `sin(0)` ঠিক পৱেৰ স্টেপে ৯০ ডিগ্ৰিৰ জন্য বসাতে হবে এবং অনুৱাপভাৱে প্ৰতি ৯০ ডিগ্ৰি অপৰ অপৰ `sin(theta)` এৰ মান প্ৰট কৰতে হবে। তাহলে গ্ৰাফটি হবে এমন (ধৰন আপনি সাইন কাৰ্ড সম্পৰ্কে কিছুই জানেন না তাই পয়েন্টগুলো সৱলৱেখা দিয়ে যোগ কৰে দিলেন)

সাইন ওয়েভ (৯০ ডিগ্ৰি পৰ পৰ প্ৰট কৰা হয়েছে)



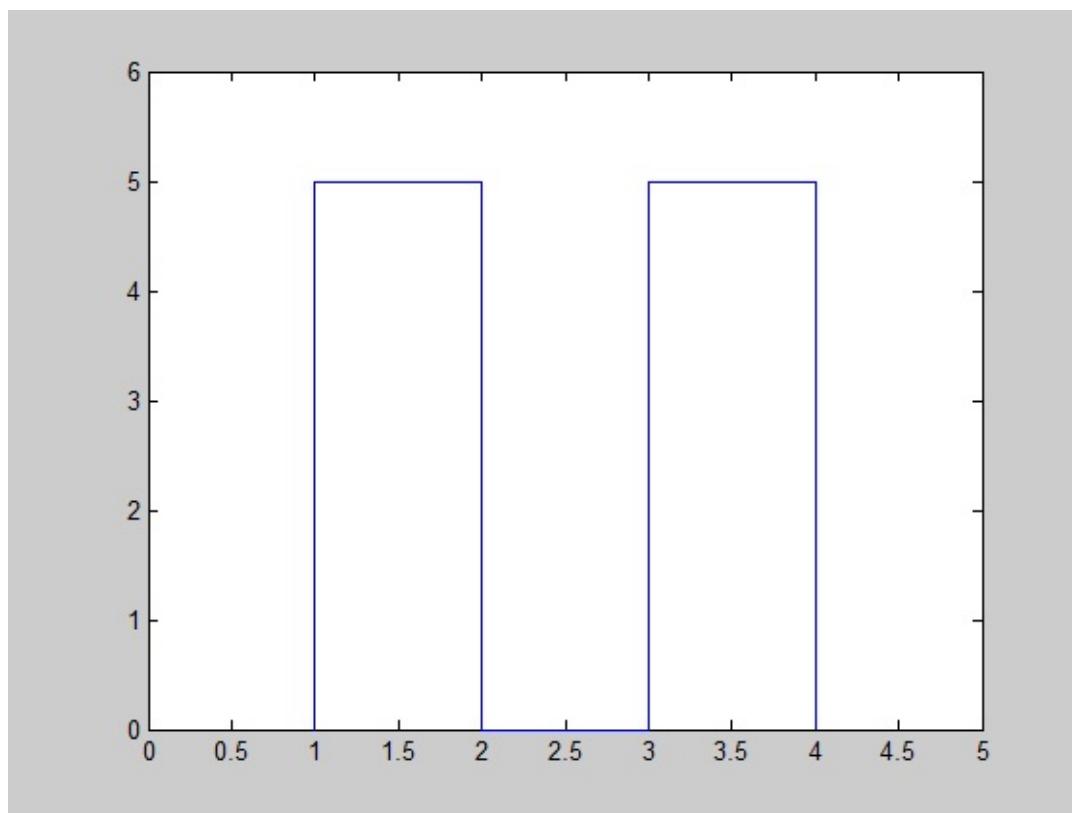
```
x = 0:90:720;
plot(x, sind(x));
```

আমরা এই গ্রাফের জন্য স্টেপ $\frac{90}{8}$ কে একক হিসেবে নিয়েছি। যেহেতু $\frac{90 \times 8}{8} = 90$ তাই একটি সাইকেলের গ্রাফ আঁকতে আমার নেওয়া স্থানাংকের পরিমাণ 8! এখন প্রতি $\frac{90}{8}$ ডিগ্রি পর পর না করে এক কাজ করা যাক, প্রতি ডিগ্রি কে একক ধরে আমরা গ্রাফটি আবার আঁকি।



ইয়েস, আগের মত Sawtooth গ্রাফ নয় বরং বেশ স্মৃথ একটি গ্রাফ পাওয়া গেল! এটা পিটের সাইন ওয়েভ না কিন্তু কাজ করার মত যথেষ্ট। সাইন ওয়েভ অ্যানালগ সিগনালের উৎকৃষ্ট উদাহরণ।

এবার চলুন একটি **ডিজিটাল সিগনাল** দেখা যাক:



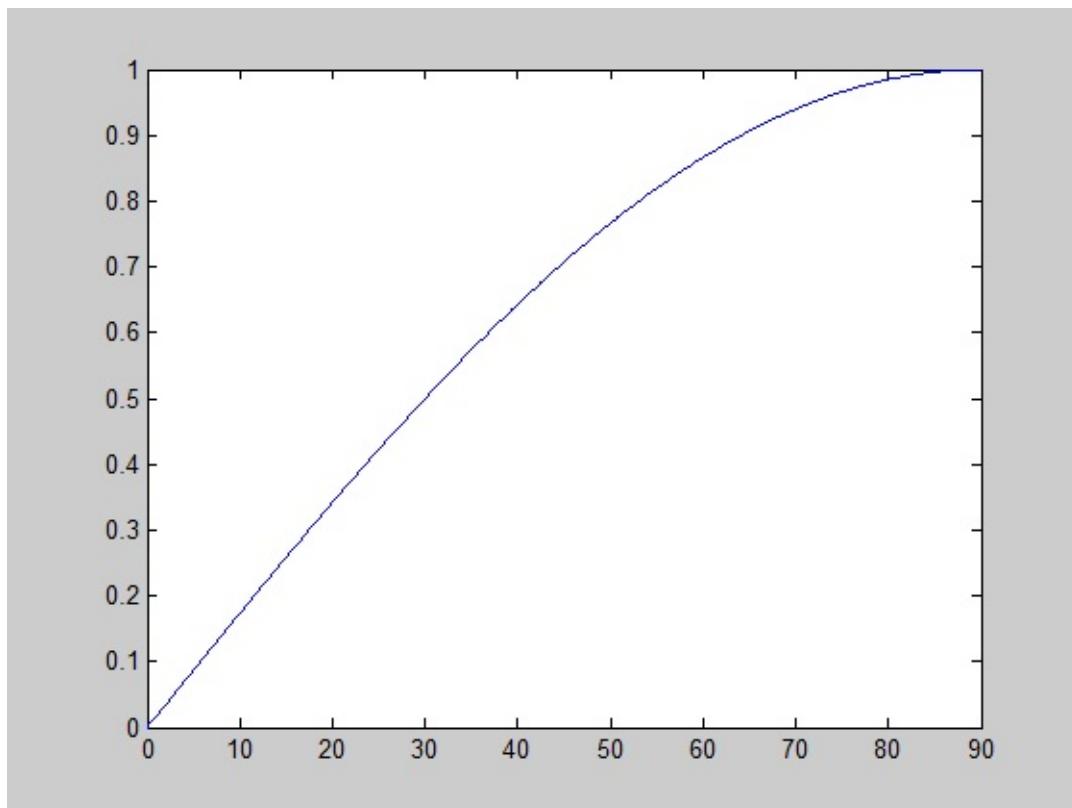
অ্যানালগ সিগনাল ও ডিজিটাল সিগনালের মধ্যে কিছু তফাহ:

অ্যানালগ সিগনাল সময়ের ক্ষুদ্রাতিক্ষুদ্র পরিবর্তনের সাথে পরিবর্তিত হয়, তাই এর বিভিন্ন সময়ে মান বিভিন্ন হয় আর আলাদা করা কঠিন; কিন্তু ডিজিটাল সিগনাল একটা নির্দিষ্ট সময় পর্যন্ত নির্দিষ্ট মান দেয় তাই আমরা সহজেই বলতে পারি (ছবি থেকে) 0-> সেকেন্ড পর্যন্ত ডোক্টেজ ০ আর >-২ সেকেন্ড ডোক্টেজ ৫।

ওই কারণেই আমরা অ্যানালগ সিগনালের ক্ষেত্রে বলতে পারি এর মান একটি নির্দিষ্ট রেঞ্জে পরিবর্তিত হয় কিন্তু ডিজিটালে যেহেতু পরিবর্তনটাও নির্দিষ্ট তাই ডিজিটালের ক্ষেত্রে 0V কে OFF বা ০ এবং 5V বা Reference Voltage কে ON ধরা হয়।

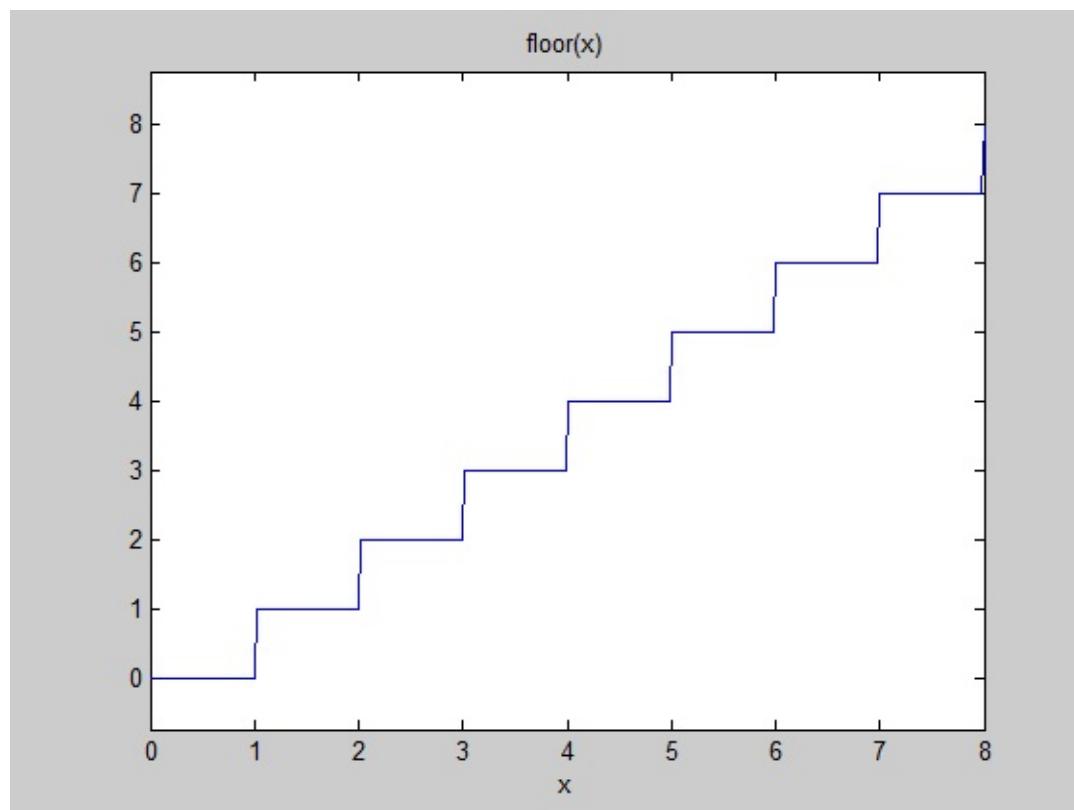
সবই বুঝলাম, কিন্তু ADC আসলে কী?

আমরা অ্যানালগ সিগনাল দেখলাম আর ডিজিটাল সিগনাল দেখলাম, ADC এর কাজ হল অ্যানালগ সিগনাল থেকে ডিজিটাল সিগনালে রূপান্তরিত করা। ডিজিটাল সিগনাল থেকে আমরা নির্দিষ্ট কিছু মান পাই সেটা আগেই দেখানো হয়েছে। ডিজিটাল সিগনালে কনভার্ট করার মাধ্যমে আমরা সেন্সরের পাঠানো অ্যানালগ সিগনাল থেকে অ্যাকচুয়াল রিডিং নিতে পারি। আবারও আমরা একটি অ্যানালগ সিগনাল দেখি:



```
theta = 0:.1:90;
plot (theta, sind(theta));
axis([0 90 0 1]);
```

উপরের গ্রাফটি হল বিখ্যাত `sine` এর। যেটা $0\text{-}90$ ডিগ্রি এর বিপরীতে মান বের করে প্লট করা হয়েছে। উপরের সিগনালটি যদি আমি `3-bit ADC Channel` এর মাধ্যমে `Pass` করি তাহলে আউটপুট আসবে নিচেরটা:



```
syms x;
sqr = floor(x);
ezplot(sqr, [0, 8]);
```

ছবি থেকে দেখা যাচ্ছে সিগনালটিকে 8 ভাগ করা হয়েছে! তারমানে 3-bit ADC Channel = $2^3 = 8$ ভাগ বা স্টেপস।

আরেকটু ক্যালকুলেশন করা যাক, যদি আমরা Peak Value বা অ্যানালগ সিগনালের সর্বোচ্চ ভ্যালু 5V ধরি তাহলে প্রতিটি ভাগ $5/8 = 0.625V$ নির্দেশ করে। তারমানে 2.5V ভোক্টেজ মাপা গেলে ADC Value হবে 4 এবং 5V রিডিং পাওয়া গেলে ADC Value হবে 8।

এই ADC Value এর Accuracy নির্ভর করে ADC এর Resolution এর উপর। আমরা আগেই দেখেছি, অ্যানালগ সিগনালকে যত ভাগ করতে পারব ততটাই নিখুঁত কাজ আমরা করতে পারব। আড়ুইনো আমাদের কতটা ফ্রেক্বিলিটি দেয় ADC এর জন্য? – আড়ুইনোর ADC এর রেজোল্যুশন 1024 তার মানে এটি 5V এর জন্য 1023 রিডিং দেবে (ইনডেক্স 0 থেকে শুরু হয় তাই 1023)। 2.5V এর জন্য রিডিং পাবেন 512.. ইত্যাদি। Arduino এর ADC Channel কে তাই 10-bit বলা হয় [$2^{10} = 1024$]।

আড়ুইনোতে ADC [অ্যানালগ টু ডিজিটাল কনভার্সন]:

আড়ুইনোতে সেন্সর রিডিং নেয়ার জন্য UNO ও Mega এর জন্য বোর্ডের বামপাশের A0-A5 (UNO) ও A0-A15(Mega) পিনগুলো ব্যবহার করা হয়। সেন্সর কানেক্ট করে আমরা সিরিয়াল মনিটরে সেন্সর রিডিং চেক করতে পারব।

ADC এর জন্য ব্যবহৃত Arduino Function :

```
int sensorValue = analogRead(int sensorPin);
```

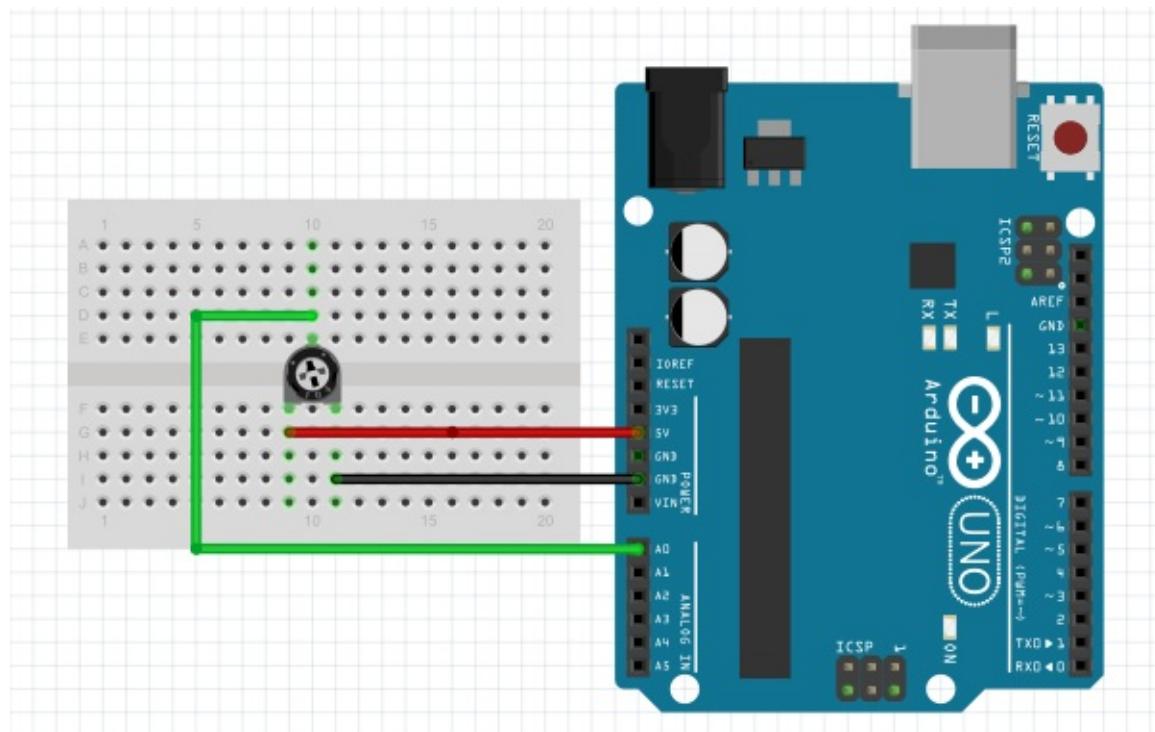
`analogRead` ফাংশনটির আগুমেন্ট ও রিটার্ন টাইপ ইন্টিজার। আগুমেন্ট হিসেবে এটি পিন নাম্বার নেয় এবং ড্যালুর রিটার্ন করে।

আডুইনো ও পটেনশিওমিটার:

পটেনশিওমিটার একটি ড্যারিয়েবল রেজিস্ট্যান্স ছাড়া কিছুই না। আমরা এর একপাশে 5V ও আরেকপাশে GND দিয়ে যদি স্ক্রু ড্রাইভার দিয়ে ঘুরাই তাহলে মাঝের পিনটি যে অংশে যুক্ত থাকবে সেটার ডেল্টেজ সেখানে থাকা জাম্পারের মধ্য দিয়ে আডুইনোতে রিড হবে। রেসিস্টিভিটি সেমবের একটি গুরুত্বপূর্ণ অংশ, টাচস্ক্রিন কিংবা জয়ষ্ঠিক মূলনীতি এই ড্যারিয়েবল রেজিস্ট্যান্স। IR, LDR এগুলো Photosensitive / Photo reflective sensor তারমানে আলোর উপর নির্ভর করে এইসব সেমব কাজ করে থাকে।

সার্কিট ডায়াগ্রাম:

খুবই সহজ কাজ, একটি পটেনশিওমিটার নিন আর তিনটা জাম্পার নিন। ছবির মত করে কানেক্ট দিন ও আডুইনো কোড আপ্লোড করুন:



প্রোগ্রাম:

ওয়াকফু-র কিছু নেই আপাতত, আডুইনো সম্পর্কিত আগের পরিচ্ছদগুলোতে ব্যাখ্যা দেওয়া আছে।

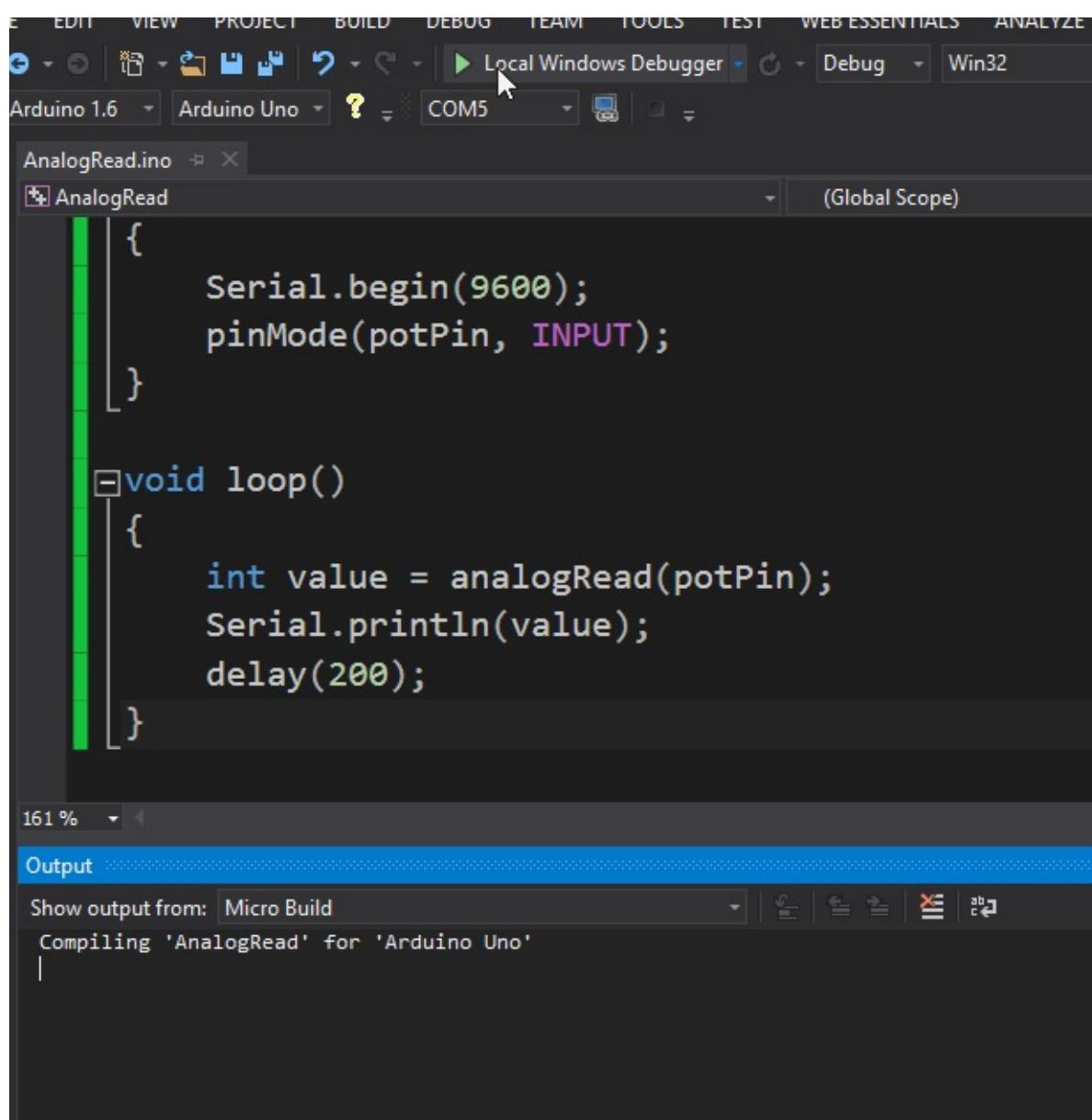
```
const int potPin = 14; //Change 14 to 53 If you use Arduino Mega

void setup()
{
    Serial.begin(9600);
    pinMode(potPin, INPUT); //Sensors are always set as INPUT
}

void loop()
{
    int value = analogRead(potPin);
    Serial.println(value);
    delay(200);
}
```

আউটপুট:

সিরিয়াল মনিটর ওপেন করুন ও একটি স্ক্রু ড্রাইভার দিয়ে পটেনশিওমিটারটিকে ঘুরিয়ে দেখুন

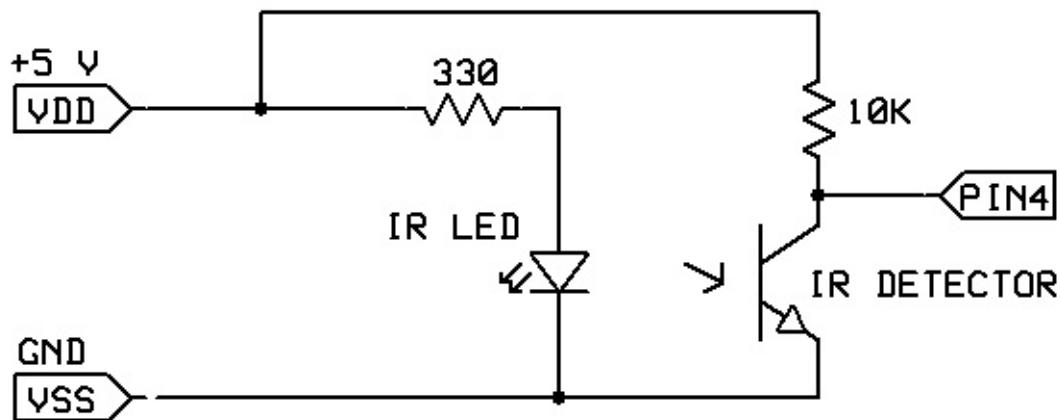


IR Sensor ও আডুইনো:

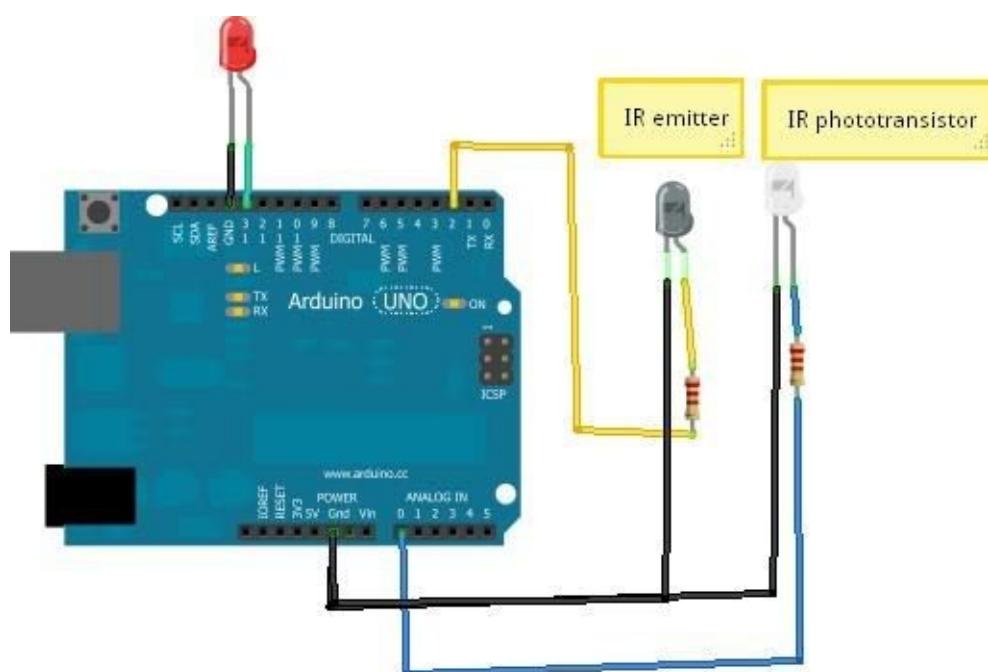
IR সেন্সর চেকিং: সেন্সর কানেক্ট করে মোবাইল বা যেকোন ক্যামেরা দিয়ে দেখুন IR Led টা জ্বলছে কিনা। IR Led এর কাজ হল IR Ray পাঠানো তাই একে IR TX বলা হয় আর IR RX এর কাজ হল IR গ্রহণ করে ডোল্টেজ দেওয়া। আমরা তাই আডুইনো পিন IR RX এর সাথে কানেক্ট করে রিডিং চেক করে থাকি। লাইন ফলোয়ার কিংবা লাইন মেজ সল্ভার বানাতে গেলে অবশ্যই আপনাকে IR সম্পর্কে ডাল করে জানতে হবে।

সার্কিট ডায়াগ্রাম:

330 Ohm রেজিস্ট্যান্স ব্যবহার করতে হবে IR TX এর সাথে এবং 10K ব্যবহার করতে হবে IR RX এর সাথে। IR Pair না থাকলে একই কাজটি আপনি LDR দিয়েও করতে পারেন। দুটির কানেকশন ডায়াগ্রাম ও প্রোগ্রাম একই। যেখানে IR LED এর বদলে ব্যবহার করবেন সাধারণ LED।



IR EMMITTER/DETECTOR HOOKUP



প্রোগ্রাম:

পটেনশিওমিটারের ক্ষেত্রে যে প্রোগ্রাম ব্যবহার করা হয়েছে সেটাতেই কাজ হবে। পরবর্তীতে সেন্সর বিডিঃ চেক করার জন্য নতুন লাইব্রেরির ব্যবহার দেখানো হবে।

সচরাচর জিজ্ঞাস্য প্রশ্ন:

ADC কী অনেকটা PWM এর মত?

উত্তর:

হ্যাঁ অনেকটা, পার্থক্য হল ডোল্টেজ দেওয়ার সময় আমরা 0-255 ভাগে ভাগ করতে পারি কিন্তু রিড করার সময় আমরা 0-1023 ভাগে ভাগ করতে পারি।

IR এ ম্যাঞ্চিমাম কত ডোল্টেজ দেওয়া যাবে?

উত্তর:

৫ ডোল্টের বেশি দিলে পুড়ে যাওয়ার সম্ভাবনা অত্যধিক। IR Led এর সাথে 330Ohm আর Receiver এর সাথে 1/10K ব্যবহার করতে ভুলবেন না।

গ্রাফগুলো কীভাবে জেনারেট করেছেন? প্রতিটি গ্রাফের শেষের কোডগুলি কিসের?

গ্রাফগুলো MATLAB দিয়ে জেনারেট করা হয়েছে, গ্রাফগুলো জেনারেট করার জন্য যে কোড ব্যবহার করেছি সেটা।

ইন্টারফেস

LCD (Liquid Crystal Display) ইন্টারফেস (প্রথম অংশ)

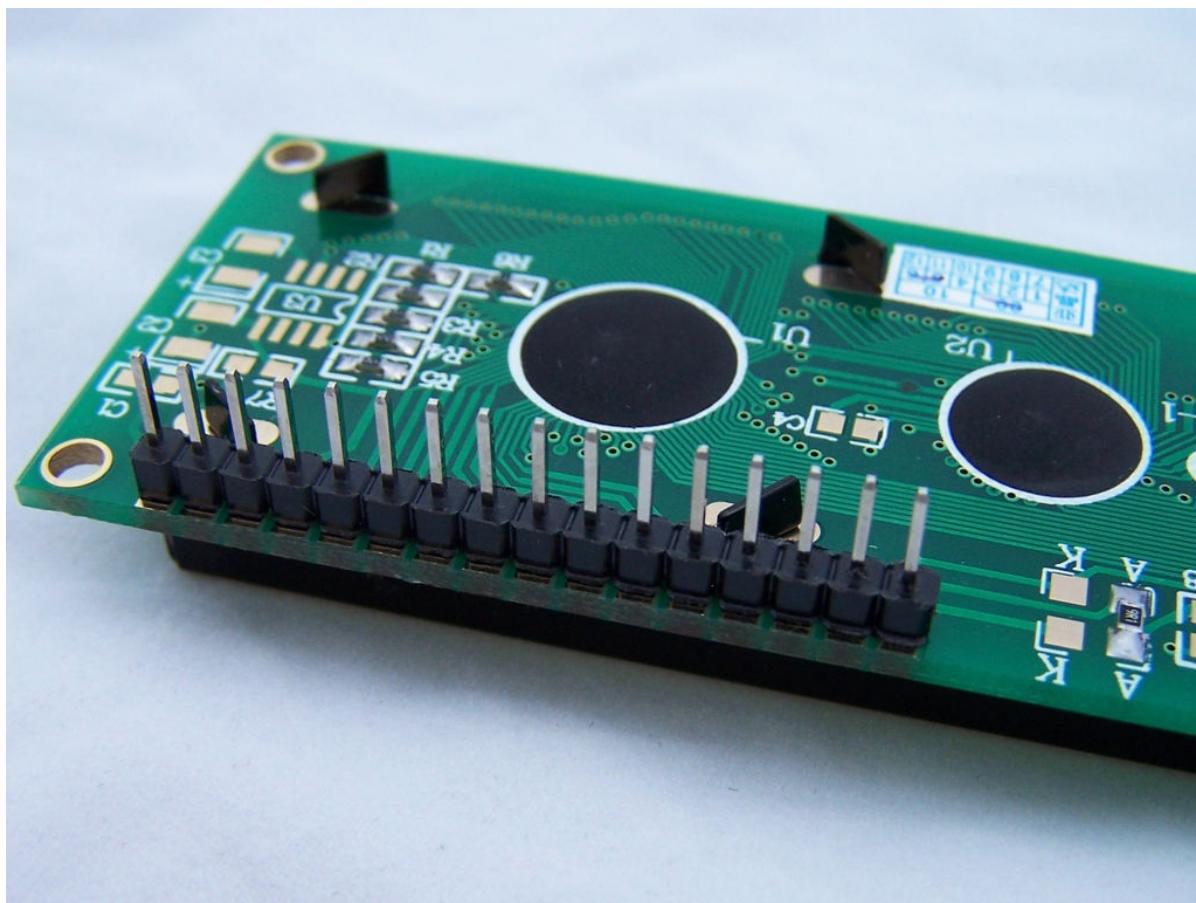
LCD খুবই কমন একটি মডিউল যেটা আমরা প্রায় সবক্ষেত্রে ব্যবহাত হতে দেখি। ডিজিটাল ঘড়ি, ক্যালকুলেটরে ইত্যাদি অনেক ক্ষেত্রেই ইনফরমেশন দেখানোর জন্য ব্যবহার করা হয়। Alphanumeric ও Graphical LCD আমরা মাইক্রোকন্ট্রুলার বা আডুইনো দ্বারা কন্ট্রুল করতে পারি। সবার সুবিধার্থে LCD কে কয়েকটি ভাগে উপস্থাপন করা হবে। আজকে আমরা দেখব কীভাবে একটি অলফানিউমেরিক্যাল LCD আডুইনোর সাথে Hookup করে এবং সাধারণ কোন লেখা ডিস্প্লে তে শো করে।

এই পর্বে যা যা লাগবে:

প্র্যাটিক্যালি দেখার জন্য

- আডুইনো (বোর্ড ও IDE)
- USB Cable (A to B)
- 16x2 / 16x4 / 20x4 .. Alphanumeric Liquid Crystal Display (LCD)

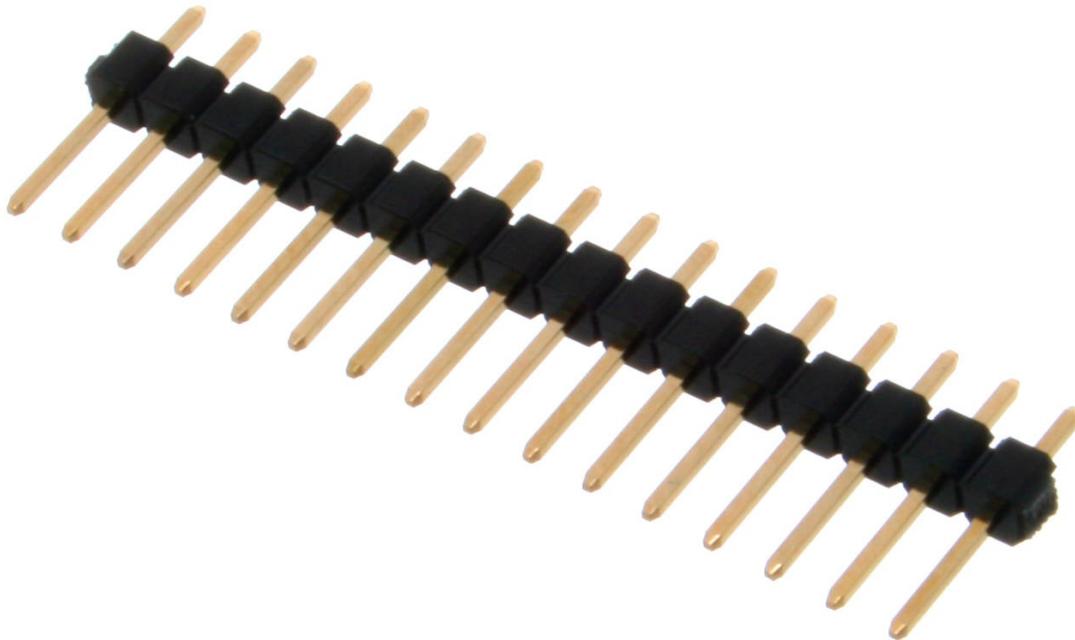




- 10k Ohm Pot
- Jumper Wires (22AWG or Premium)
- Breadboard

| যদি আপনার LCD তে Header Pin Soldered করা না থাকে তাহলে

- Soldering Iron
- সীসা
- রঞ্জন
- Header Pin



ডার্চয়ালি দেখার জন্য

- Proteus ISIS
- Arduino IDE

LCD সেট আপ:

Alphanumeric LCD গুলো খুবই কমন, এগুলোর ১৬ টি করে পিন থাকে। ১৪ টি পিন থাকার মানে হল Backlight নেই। অর্থাৎ ১৪ টি পিন + ২ টি ব্যাকলাইটের জন্য। আপনার LCD তে কোন হেডার পিন না থাকলে আপনার নিজেরই সেটা সোন্দার করে নিতে হবে। পর্বের শেষের দিকে সোন্দারিং ইন্স্ট্রুকশন দিয়ে দেওয়া হল।

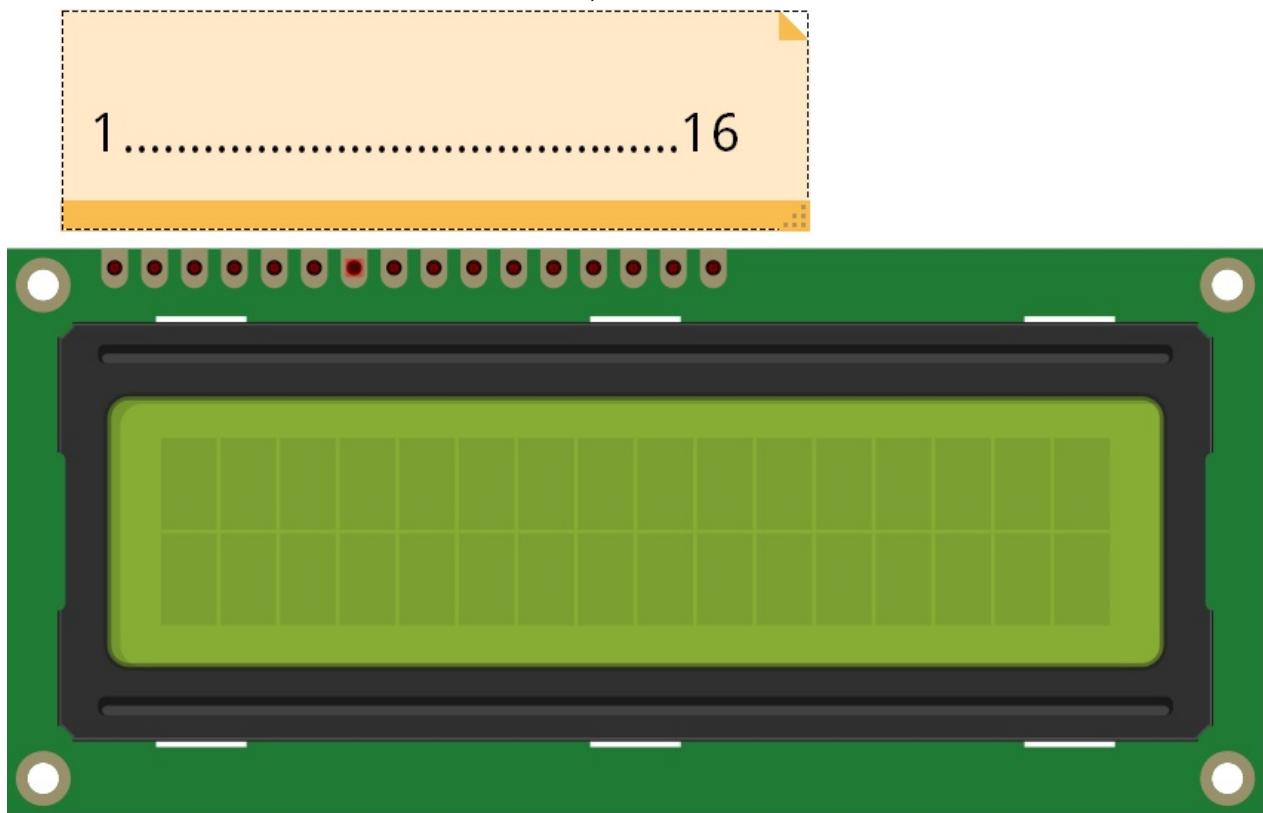
হেডার পিন আছে বা নেই বুঝতে পারছেন না? ছবির LCD টি দেখুন, তাহলেই বুঝতে পারবেন।

গুরুত্বপূর্ণ: আপনার **LCD** এর ডাইমেনশন কাজ শুরুর আগেই জেনে নিন, অর্থাৎ এটা কী **16x4**, **16x2** নাকি **20x2** বা **20x4**, বুঝতে না পারলে LCD টা কে আলোয়ে ধরে যে কয়টা কাল ব্লকের কলাম আর রো গণনা করুন তাহলেই পেয়ে যাবেন।

এইবার LCD কে আডুইনোর সাথে wiring করতে হবে। LCD module এর সেম পিন দিয়ে দুই ধরণের wiring করা যায়। একটি হল 4-pin আরেকটি 8-pin mode। সব কিছু সিম্পল রাখার সুবিধার্থে আমরা 4-pin মোডেই কাজ করব। LCD পিন আউট টেবিলটি খুব কাজে লাগবে কাজেই সেটা একবার দেখে নেওয়া ভাল:

LCD পিন আউট ডায়াগ্রাম:

LCD এর পিন নামার কাউন্টিং শুরু হয় বামপাশ থেকে, নিচের ছবি দেখলেই পরিষ্কার হবে



fritzing

LCD পিনআউট টেবিল

পিন নাম্বার Pin no.	পিনের নাম Pin Name	যে কাজে ব্যবহার করা হবে Pin Purpose
1	VSS	GND Connection
2	VDD	+5V Connection
3	V0	Contrast Adjustment
4	RS	Register selection (Character vs Command)
5	RW	Read/Write
6	EN	Enable
7	D0	Data Line 0 (Unused)
8	D1	Data Line 1 (Unused)
9	D2	Data Line 2 (Unused)
10	D3	Data Line 3 (Unused)
11	D4	Data Line 4
12	D5	Data Line 5
13	D6	Data Line 6
14	D7	Data Line 7
15	A	Backlight Anode (+5V Connection)
16	K	Backlight Cathode (GND Connection)

পিন কানেকশন ব্যাখ্যা

V0 [3rd Pin]:

- V0 পিনের কাজ হল LCD এর কন্ট্রাস্ট Contrast কন্ট্রোল করা। এর জন্য আমাদের একটি 10KOhm Pot ব্যবহার করতে হবে। এই পট V0 বা LCD এর 3 নাম্বার পিনের সাথে কানেক্টেড থাকবে। পটের মাধ্যমে রেজিস্ট্যাম্প পরিবর্তন করলে কন্ট্রাস্ট ও পরিবর্তিত হবে।
 - উল্লেখ্য V0 কে যদি আমরা গ্রাউন্ডে কানেক্ট করি তাহলে ফুল কন্ট্রাস্ট পাওয়া যাবে।

RS [4th Pin]:

- এই পিনের কাজ হল LCD কে Command বা Character mode এ সেট করা। এটা দ্বারা LCD বুঝতে পারে যে Data Line এ আসা Data set কে সে কীভাবে কাজে লাগাবে। সেটি কি কোন কমান্ড নাকি ক্যারেক্টার?

RW [5th Pin]:

- এই পিনটিকে সবসময়ই আমরা গ্রাউন্ডে কানেক্ট করে রাখব। মানে LCD তে আমরা সবসময় Write করব।

EN [6th Pin]:

- EN পিনের মাধ্যমে LCD কে Data Line এর স্ট্যাটাস জানানো হয়।

Backlight Anode & Cathode [15th & 16th Pin]:

- আগেই বলা হয়েছে এই পিনদুটো সব LCD তে নাও থাকতে পারে। যদি থাকে তাহলে 15 কে `+5V` এর সাথে কানেক্ট করুন ও 16 কে `GND` তে।

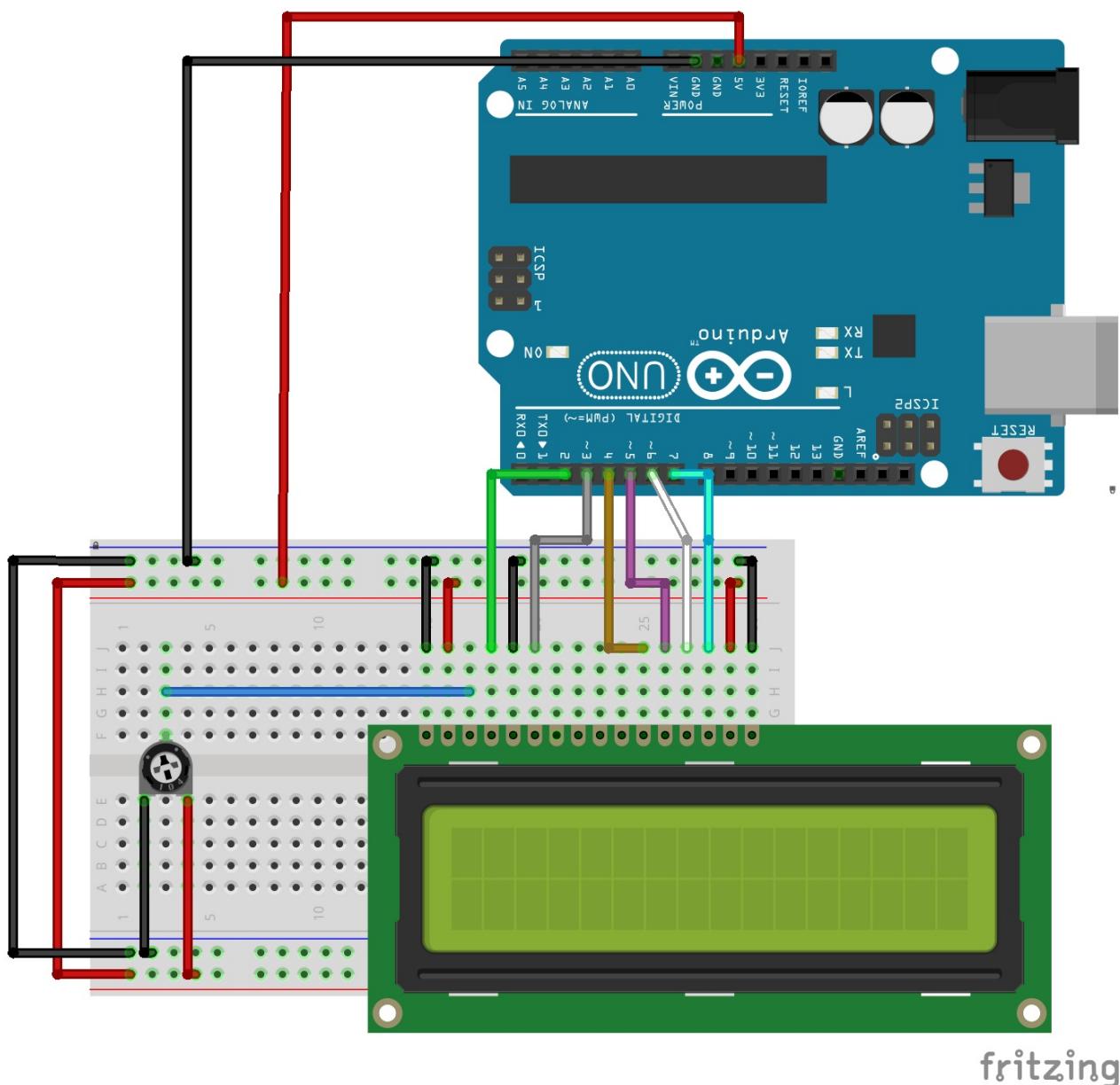
কম্যুনিকেশন পিন কানেকশন

LCD কম্যুনিকেশনের জন্য আমরা 6 টা পিন ব্যবহার আডুইনোর নিচের পিনগুলোর সাথে কানেক্ট করব:

LCD Pin	Arduino Pin Number
RS	Pin 2
EN	Pin 3
D4	Pin 4
D5	Pin 5
D6	Pin 6
D7	Pin 7

সার্কিট ডায়াগ্রাম

Arduino ও LCD কে নিচের ছবির মত করে কানেক্ট করে ফেলুন



fritzing

প্রোগ্রাম

যেহেতু আমরা কেবল LCD কানেকশন দিয়েছি ও টেস্ট করতে চাই, তাই আমরা সিম্পল কোড দিয়ে শুরু করব।
এরপর আমরা আস্তে আস্তে জটিলের দিকে যাব। নিচের প্রোগ্রামটি LCD তে "Electroscholars" শো করবে আপনি
সেখানে আপনার নাম বসিয়েও টেস্ট করে দেখতে পাবেন।

```
//Importing LiquidCrystal Library for driving the LCD

#include <LiquidCrystal.h>

//LCD Dimension
#define COLS 16
#define ROWS 4

//Creating global LCD Object
LiquidCrystal my_lcd(2, 3, 4, 5, 6, 7);

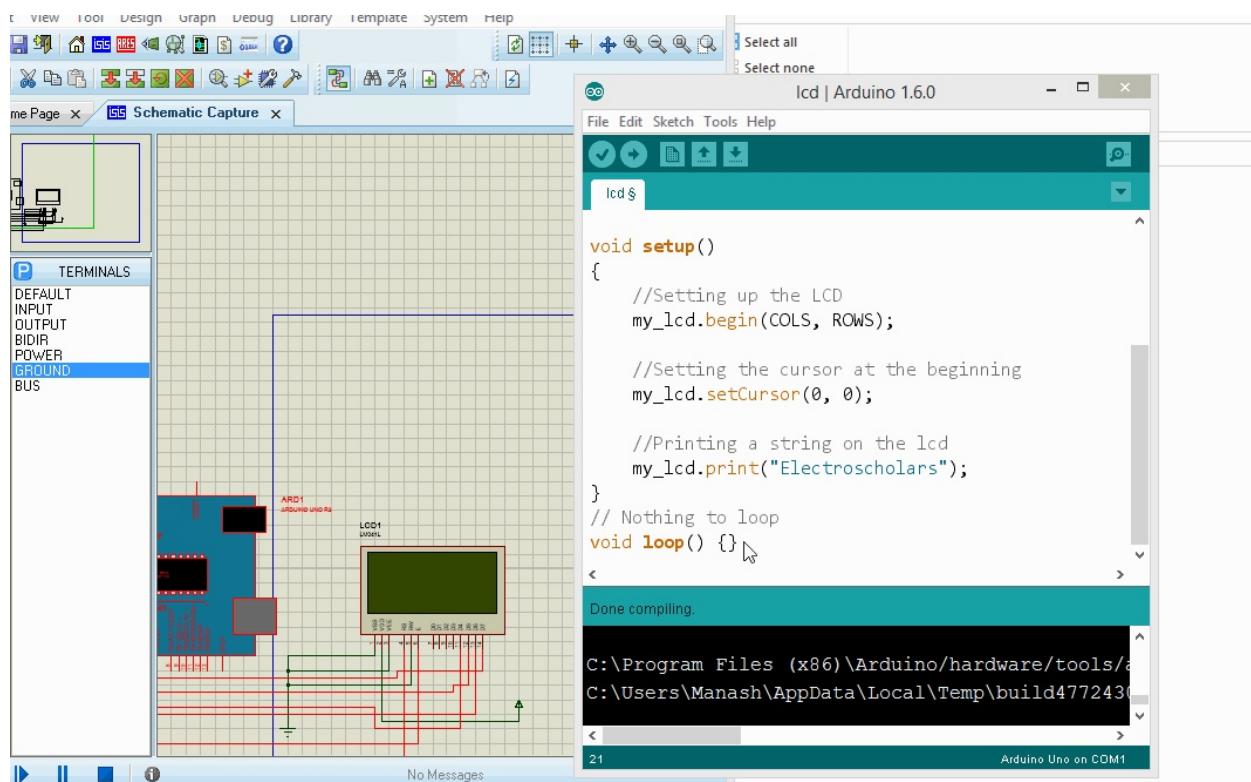
void setup()
{
    //Setting up the LCD
    my_lcd.begin(COLS, ROWS);

    //Setting the cursor at the beginning
    my_lcd.setCursor(0, 0);

    //Printing a string on the lcd
    my_lcd.print("Electroscholars");
}

//Nothing To Loop
void loop() {}
```

আউটপুট (প্রোটিয়াস সিমুলেশন):



প্রোগ্রাম ওয়াকচুন

```
#import <LiquidCrystal.h>
```

LCD চালানোর জন্য আডুইনোর অফিশিয়াল LiquidCrystal লাইব্রেরি আমরা ব্যবহার করব। লাইব্রেরি ব্যবহার করার মজা হল আপনি সরাসরি অন্য প্রোগ্রামারদের কোড কোড আপনার কাজে ব্যবহার করতে পারবেন।

LiquidCrystal লাইব্রেরিতে প্রচুর ফাংশন তৈরি করে দেওয়া আছে, যেমন কার্সর রিংকিং, কাস্টম ক্যারেষ্টার তৈরি করা, টেক্সট ডিরেকশন পরিবর্তন করা (left to right / right to left) ইত্যাদি। এই পর্বের শেষে আমরা অ্যাডেইল্যাবল ফাংশনগুলো ও তাদের প্রয়োগ সম্পর্কে হাস্তা ধারণা নেব।

আডুইনোর প্রাণ মূলত আডুইনো লাইব্রেরিগুলোই, যা ওপেন-সোর্স; তারমানে আপনি লাইব্রেরি গুলো মডিফাই করে নিজের কাজে ব্যবহার করতে পারবেন ও মডিফাই করা কোড আবার অন্যদের মাঝে বিতরণ করতে পারবেন। পরবর্তী কয়েকটি পর LCD নিয়েই করা হবে এবং এই লাইব্রেরিটাই ব্যবহার করা হবে। লাইব্রেরি মডিফাই করে নিজের লাইব্রেরি তৈরি করাও আমরা দেখব।

```
#define ROWS, #define COLS
```

LCD নিয়ে কাজ করতে হলে অবশ্যই ডাইমেনশন জানতে হবে। মানে আপনার LCD কয়টি ক্যারেষ্টার সাপোর্ট করে। `#define` করা ভ্যারিয়েবল গুলো আর ভ্যারিয়েবল থাকে না, ওটা কনস্ট্যাট হয়ে যায়। যেহেতু আমাদের LCD এর রো আর কলাম সংখ্যা ফ্রেক্ষন তাই সেটাকে variable হিসেবে ডিক্রিয়ার না করাই ভাল। ইচ্ছা করলে রো আর কলাম আমরা এভাবেও লিখতে পারতাম:

```
const int rows = 4;
const int cols = 16;
```

অথবা সরাসরি কলাম আৰ বো বসিয়ে দিলেও হত

```
my_lcd.begin(16, 4);
```

LiquidCrystal my_lcd(. . .)

অবজেক্ট ওরিয়েন্টেড প্ৰোগ্ৰামিং সম্পর্কে আমৰা আগেই জেনে ছিলাম। এবাৰ সেটা প্ৰয়োগ কৰাৰ সময় চলে এসেছে। কনষ্ট্ৰাইটৱেৰ কথা মনে আছে তো? না মনে থাকলে [একবাৰ দেখে নিন।](#)

`LiquidCrystal` হল ক্লাস যাৰ ডেফিনিশন দেওয়া আছে `#LiquidCrystal.h` নামক হেডাৰ ফাইলটিতে। আৰ অবজেক্ট তৈৰি কৰতে হয় এভাবে:

```
//Without constructor arguments
the_class objectName;

//If the constructor takes arguments
the_class objectName(1, 2, 3);
```

`LiquidCrystal` এ ক্লাসেৰ Constructor ওভাৱলোডেড। কোনটা ইউজ হবে সেটা নিৰ্ভৰ কৰে আপনাৰ পাঠানো আণ্টমেটোৰ উপৰে। আমৰা LCD এৰ ৬ টা পিন আডুইনোৰ I/O এৰ সাথে কানেক্ট কৰেছি তাই ৬ আণ্টমেট বিশিষ্ট কনষ্ট্ৰাইটৱেৰ কাজ কৰবে। ৬ টা পিন অবশ্যই ক্ৰমানুসাৰে পাঠাতে হবে। যেমন এই প্ৰোগ্ৰামেৰ ক্ষেত্ৰে ব্যবহৃত Constructor এৰ পিনেৰ ক্ৰম ছিল

```
LiquidCrystal lcd(RS, EN, D4, D5, D6, D7);
```

খেয়াল কৰে দেখুন আমৰা এই ক্ৰম অনুযায়ী পিনগুলো কানেক্ট কৰেছি ও পিনগুলোৰ তথ্য কনষ্ট্ৰাইটৱেৰ মাধ্যমে পাঠাচ্ছি।

my_lcd.begin(COLS, ROWS);

LCD কে ক্যারেষ্টাৰ ডিস্প্ৰেছ কৰাৰ জন্য ইনিশিয়ালাইজ কৰতে হবে। তাই আমৰা `LiquidCrystal::begin(const uint8_t, const uint8_t)` মেথডেৰ মাধ্যমে ব্যবহৃত LCD এৰ ডাইমেনশন পাঠিয়ে দিলাম সব সেট কৰাৰ জন্য।

যেহেতু LCD টি কমান্ড গ্ৰহণ ও ক্যারেষ্টাৰ শো কৰাৰ জন্য প্ৰস্তুত তাই আমৰা এবাৰ

`LiquidCrystal::print(char* str)` ৩ `LiquidCrystal::setCursor(uint8_t x, uint8_t y)` মেথড দুটো ব্যবহাৰ কৰতে পাৱাৰ।

my_lcd.setCursor(x, y)

এই কমান্ডের মাধ্যমে Cursor কোথায় সেট করতে হবে সেটা ঠিক করা যায়। যদি আমি কোন একটি লেখা LCD এর ত্বরণ সারির ২য় কলাম থেকে স্টার্ট করতে চাই তাহলে কমান্ডটি হবে `my_lcd.setCursor(1, 2); //1 for second column, 2 for 3rd row [zero indexing]`

my_lcd.print(..)

`LiquidCrystal::print(char *str)` মেথডের মাধ্যমে আপনি LCD তে কোন লেখা শো করতে পারবেন। এর আর্গুমেন্ট স্ট্রিং টাইপ।

যেহেতু আমাদের লুপিংয়ের কোন প্রয়োজন নেই তাই আমরা `void loop()` ফাংশনটিতে কিছু লিখিনি।

যদি এখন LCD তে লেখা ভেসে উঠে তাহলে বুঝবেন আপনি সঠিকভাবে আপনার LCD টি আডুইনোর সাথে কনফিগার করতে পেরেছেন। এই পর্যন্তই, LCD এর পরবর্তী পর্যে আমরা কিছু ছোটখাট প্রজেক্ট তৈরি করব।

LCD Header পিন সোল্ডারিং

সোল্ডার করার জন্য [এই টিউটোরিয়ালটি](#) ফলো করুন। অথবা ইউটিউবে সার্চ দিলেই অনেক টিউটোরিয়াল পাবেন।

সোর্স কোড ও ডায়াগ্রাম

সোর্স কোড, ডায়াগ্রাম সবই আপ্লোড করা আছে [এখানে](#)। চাইলে নামিয়ে টেস্ট করতে পারেন।

আডুইনো লাইব্রেরি তৈরি ও ইনহেরিট্যান্স প্রয়োগের মাধ্যমে লাইব্রেরি মডিফাই করা

এই পর্বের জন্য যা যা প্রয়োজন:

- ১। একটি ভাল C/C++ IDE [Code::blocks, Codelite, Qt Creator, Visual Studio..] অথবা Text Editor [Notepad++, Sublime Text Editor, Atom, Brackets ...]
- ২। লাইব্রেরি টেস্ট করার জন্য Official Arduino IDE
- ৩। যেকোন আডুইনো বোর্ড
- ৪। বোর্ড না থাকলে সিম্যুলেশনের জন্য প্রোটিয়াস (Proteus)
- ৫। LCD Module (যেকোন ডাইমেনশনের)

লাইব্রেরি তৈরি করার যৌক্তিকতা:

আডুইনোতে যে এত সহজে কোড করতে পারছেন, শুধু digitalWrite(size_t, MODE) কিংবা
analogRead(size_t) ফাংশনগুলো ব্যবহার করলেই LED ইচ্ছামত জুলানোভা করা অথবা সেমুলেশন থেকে রিডিং
নেওয়া। Serial.read() ব্যবহার করে সহজেই সিরিয়াল কম্যুনিকেশন এস্টেলিশ করে ডিবাগিং করা ইত্যাদি
কাজগুলো সহজ হয়েছে মূলত Arduino লাইব্রেরির কারণে। Arduino আসলে একটি Avr চিপ ছাড়া কিছুই
না। এতে Communication এর জন্য Atmega8U2 / Atmega16U2 চিপ ব্যবহার করা হয়েছে এবং ব্যাটারি জনিত
সমস্যা এড়ানোর জন্য বিল্ট-ইন ভোল্টেজ বেগুলেটর* ব্যবহৃত হয়েছে। এ সব কিছু বাদ দিলে শুধু Atmega328p-PU
চিপটাই থেকে যায়। আর আমরা সবাই জেনে গেছি এটা মূলত একটি AVR চিপ। AVR-C তে করা
প্রোগ্রামগুলোকে সহজ করার জন্য Arduino Library ব্যবহার করা হয়েছে। কারণ সবাই বিটওয়াইজ অপারেশনকে
শুরুর দিকে বেশ ঝামেলার কাজ বলে মনে করে, আর তাই তাদের হার্ডওয়্যারের প্রতি আগ্রহ উভে যায়। তাই এটাকে
সহজ ও আনন্দদায়ক করার জন্য হাজারখানেক লাইনের কোড লিখে আডুইনো অফিশিয়াল লাইব্রেরি তৈরি করা হয়।

প্রোগ্রামগুলো এত সহজ হওয়ার কারণ মূলত ওই লাইব্রেরি। ওপেন-সোর্সের মজা মূলত এখানেই, নিজের ইচ্ছামত
সবকিছু ডেঙ্গে গুঁড়িয়ে আবার নতুন করে শুরু করা যায়। Code Reusability এর ক্ষেত্রে লাইব্রেরির জুড়ি নেই।

যেমন, PIC এর জন্য MikroC এর তৈরি লাইব্রেরি আপনি ইচ্ছা করলেই ব্যবহার করতে পারবেন না। এই লাইব্রেরি
ব্যবহার করার জন্য আপনাকে Pay করতে হবে। সেখানে আডুইনো লাইব্রেরি ব্যবহার করা/ মডিফাই করা কিংবা
মডিফাই করে রিডিস্ট্রিভিউট করার ক্ষেত্রে কোন বাঁধা নেই।

লাইব্রেরি তৈরি ও ব্যবহার

লাইব্রেরি তৈরি করতে ও এর সঠিক ব্যবহার করতে দক্ষতার প্রয়োজন হয়। লাইব্রেরি শুধু লিখলেই হয় না, সহজবোধ্য ভ্যারিয়েবল তৈরি, সুন্দর সিনট্যাক্স ও কমেন্টিংয়ের মাধ্যমে বোঝা যায় কোন লাইব্রেরি কতটা ভাল।

ধরা যাক, আপনি মেমরি এফিশিয়েন্ট একটি লাইব্রেরি তৈরি করলেন, কিন্তু আপনার ডকুমেন্টেশনটা যুক্তসই হল না, সেক্ষেত্রে যিনি আপনার লাইব্রেরি ব্যবহার করবে সে অনেক সমস্যার সম্মুখীন হতে পারে। কিন্তু আরেকজনের লাইব্রেরি তেমন মেমরি এফিশিয়েন্ট হলনা কিন্তু তার লাইব্রেরির ডকুমেন্টেশন, সিনট্যাক্স ও কমেন্টিংয়ের কারণে তার লাইব্রেরি ব্যবহার করা খুবই আরামদায়ক হলে সেই লাইব্রেরিটাই সবার কাছে গ্রহণযোগ্য হবে। তাই লাইব্রেরি তৈরি করার সময় আপনাকে এসব কিছু খেয়াল রাখতে হবে।

লাইব্রেরি ব্যবহার করার জন্য লাইব্রেরি লেখকের ডকুমেন্টেশন ভালভাবে বুঝার চেষ্টা করতে হবে। তিনি কোন কোন মেথড তৈরি করেছেন, কোনটা দিয়ে কি করে; কোনটার কী লিমিটেশন অথবা আপনার যদি মনে হয় আপনি সেটাকে মডিফাই করে আরও ভাল কিছু তৈরি করতে পারবেন, সেক্ষেত্রে আপনি সে লাইব্রেরি আপডেট দিয়ে তাকে জানিয়ে দিতে পারেন।

আডুইনোর কিছু ডিফল্ট ও গুরুত্বপূর্ণ লাইব্রেরির লিস্ট

আমরা আডুইনোর এই কোর্সে যেসব লাইব্রেরি ব্যবহার করব

- [SoftwareSerial](#)
- [LiquidCrystal](#)
- [Wifi](#)
- [NewPing](#)
- [SPI](#)

[ইত্যাদি](#)

আডুইনো লাইব্রেরির গঠন

ধরি আমার আডুইনো লাইব্রেরির নাম `MyLibrary`

তাহলে,

- `MyLibrary` ফোল্ডার: এই ফোল্ডারে আপনার লাইব্রেরির প্রয়োজনীয় ফাইলগুলো থাকবে
 - `examples` ফোল্ডার `[MyLibrary/examples/MyExample/MyExample.ino]`: এই ফোল্ডারে আপনার তৈরিকৃত লাইব্রেরি ব্যবহার করে কোন একটি প্রোগ্রাম থাকবে, সেটা অবশ্যই `.ino` এক্সেনশনবিশিষ্ট আডুইনো ফাইল হতে হবে। এবং সেই ফাইলটি আরেকটি ফোল্ডারে রাখতে হবে যার নাম একই।
 - `changelog.txt` `[MyLibrary/changelog.txt]`: লাইব্রেরির ডার্সন আপডেট, নতুন ফিচার অ্যাড ইত্যাদি তথ্য এই টেক্সট ফাইলে অ্যাড করতে হবে। (optional)
 - `keywords.txt` `[MyLibrary/keywords.txt]`: আপনার লাইব্রেরির Class, method, properties এ যদি Syntax Highlighting প্রয়োগ করতে চান, অথবা যদি চান আপনার তৈরিকৃত Class, method ইত্যাদি IDE রঙিন করে দিক তাহলে এই ফাইলে সেই Keyword গুলো লিখতে হবে। (optional)
 - `MyLibrary.h` `[MyLibrary/MyLibrary.h]`: এটা হল আপনার লাইব্রেরির হেডার ফাইল, হেডার

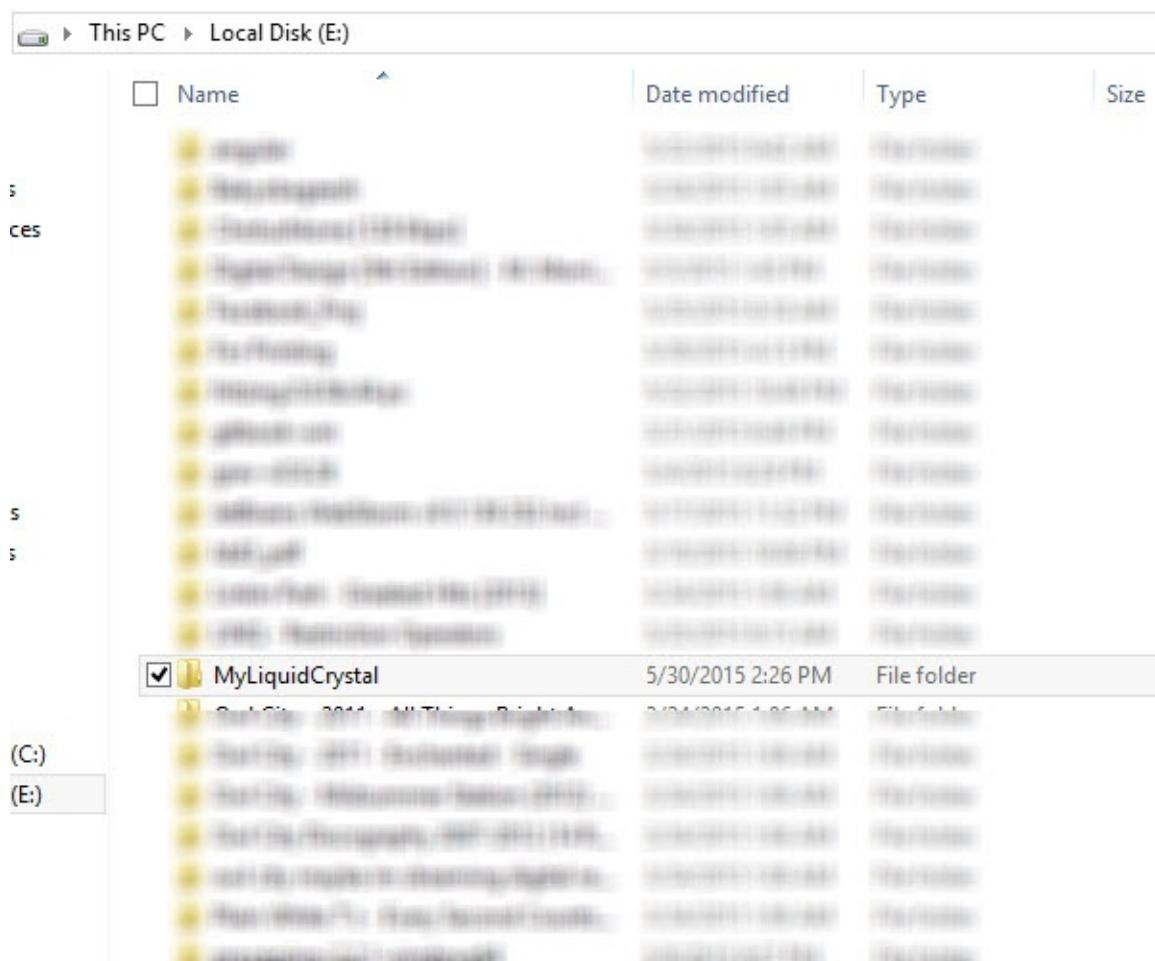
ফাইলকে আমরা মেন্যুও বলতে পারি। আপনার তৈরি ক্লাসের যাবতীয় সব তথ্য এইখানে রাখলে একনজরে দেখতে সুবিধা হয়। লাইব্রেরি ও ক্লাসের নাম একই হতে হবে এমন কোন কথা নেই, কিন্তু এক রাখাটা সুবিধাজনক। স্ট্রাকচার্ড ও স্প্লিট ফাইল বেজড প্রোগ্রামিংয়ে Header ফাইল ব্যবহার করা ভাল প্রোগ্রামিং প্র্যাক্টিস।

- MyLibrary.cpp [MyLibrary/MyLibrary.cpp] : এটা হল আপনার তৈরি করা Header ফাইলের সোর্স ফাইল। নাম একই হতে হবে এমন কোন কথা নাই, নাম এক হলে চিনতে সুবিধা হয়।
- অন্যান্য .h ও .cpp ফাইল [MyLibrary/Others.h, MyLibrary/Others.cpp]: আপনার তৈরি লাইব্রেরি যদি অন্য কোন লাইব্রেরি / ক্লাস বা মেথড নির্ভরশীল হয় তাহলে সে ফাইলগুলো আপনার লাইব্রেরির ফোল্ডারে রেখে দেবেন।

এই পর্বের লাইব্রেরির উদাহরণ:

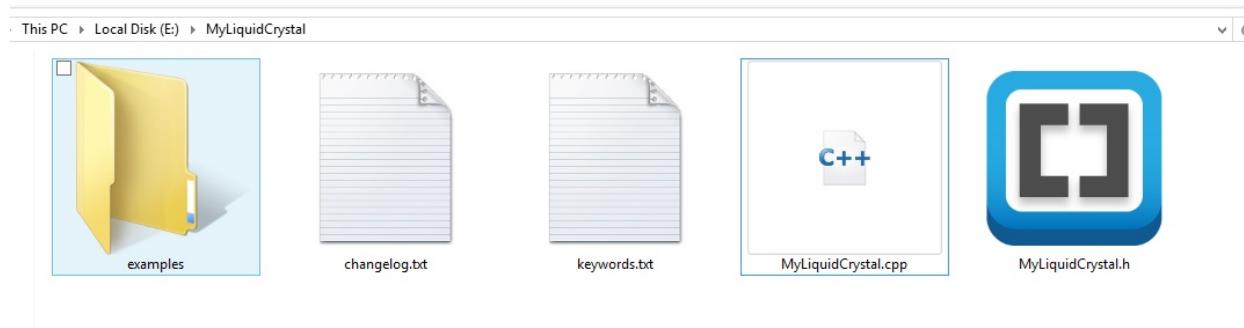
আমরা LiquidCrystal লাইব্রেরিকে মডিফাই করব, তাহলে আমাদের লাইব্রেরির জন্য যে ফাইল/ ফোল্ডারগুলো লাগবে, একনজরে দেখা যাক:

- আমি যে নামে লাইব্রেরি তৈরি করতে চাই সে নামের একটা ফোল্ডার তৈরি করলাম (লাইব্রেরি তৈরি শেষে আড্ডুইনোতে ইস্পেক্ট করব)

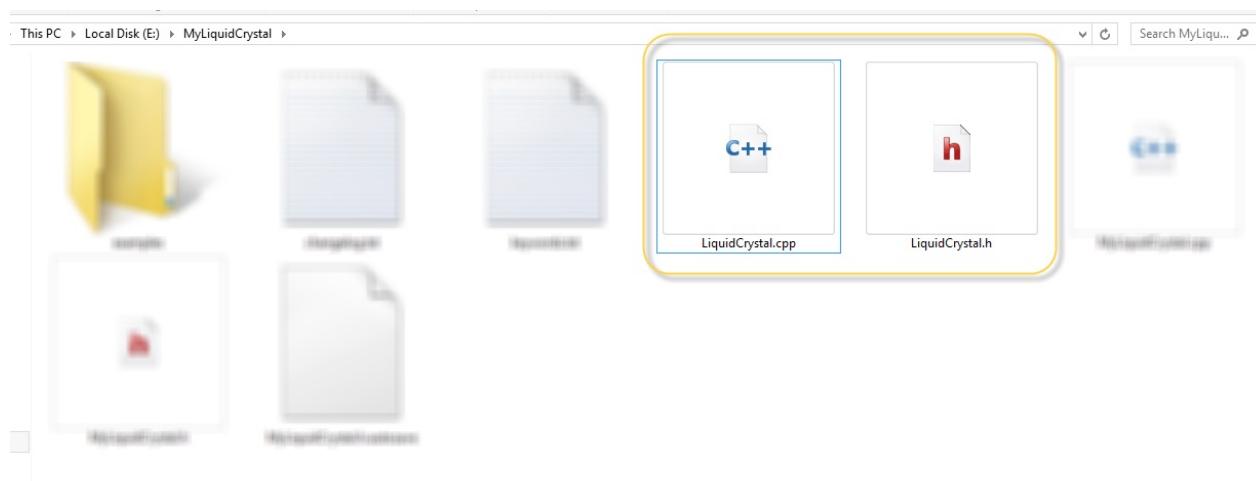


- ফোল্ডারে স্কেলিটন ফাইলগুলো তৈরি করে ফেললাম
 - MyLiquidCrystal.h
 - MyLiquidCrystal.cpp

- keywords.txt
- changelog.txt
- examples\ScrollText\ScrollText.ino



- MyLiquidCrystal লাইব্রেরিটা আডুইনো অফিশিয়াল LiquidCrystal ও Arduino.h এর উপর নির্ভরশীল। তাই LiquidCrystal লাইব্রেরির সোর্স (LiquidCrystal.h ও LiquidCrystal.cpp) ডাউনলোড করে ও Arduino.h ডাউনলোড করে আমার লাইব্রেরির ডিরেষ্টরিতে রাখলাম।



লাইব্রেরি লজিক

আগেই বলে নেই, যে লজিকটা এখানে আমি ইন্পিমেন্ট করব সেটা Buggy এবং অনেক সমস্যা আছে। এই সমস্যাগুলোর কয়েকটা আমি উল্লেখ করে দেব এবং সল্ভ করার কিছু হিন্টস দেব, তবে সমস্যাটা আপনাকেই সল্ভ করতে হবে। যদি এর সমাধান চেয়ে অনেক রেসপন্স আসে তাহলে তার সমাধান দেওয়ার চেষ্টা করব।

যা করতে চাই:

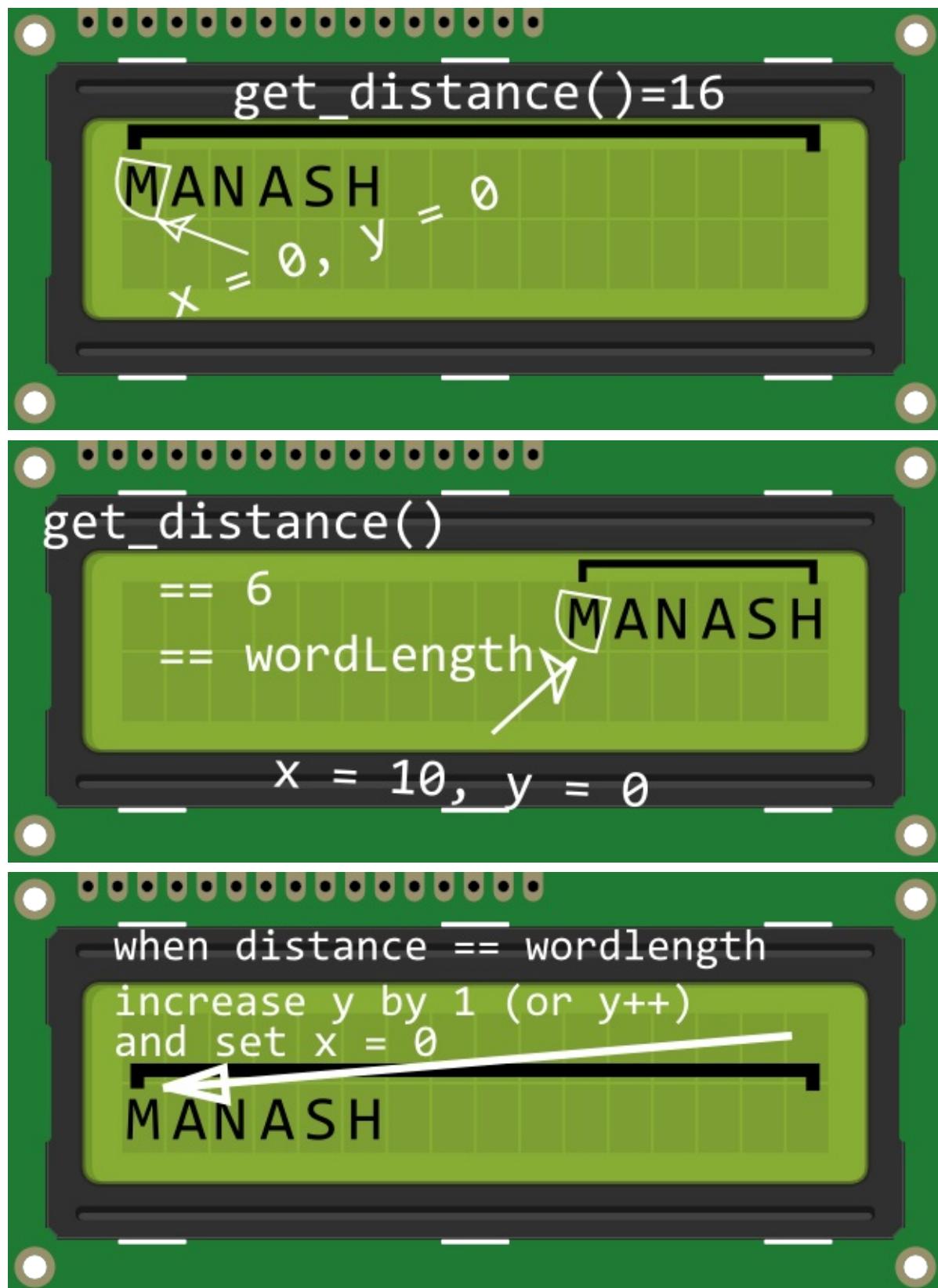
LiquidCrystal লাইব্রেরিকে মডিফাই করে আমরা একটা মেথড অ্যাড করতে চাই, যার কাজ হবে একটা নির্দিষ্ট স্ট্রিং বা শব্দকে LCD তে স্ক্রল করানো। LiquidCrystal এ অলবেডি এই মেথড তৈরি করা আছে কিন্তু সেই মেথড পুরো ডিস্প্লে স্ক্রল করতে থাকে। এইখানে আমি একটি Buggy code লিখব যেটা পরবর্তীতে আপনাকে ঠিক করতে হবে।

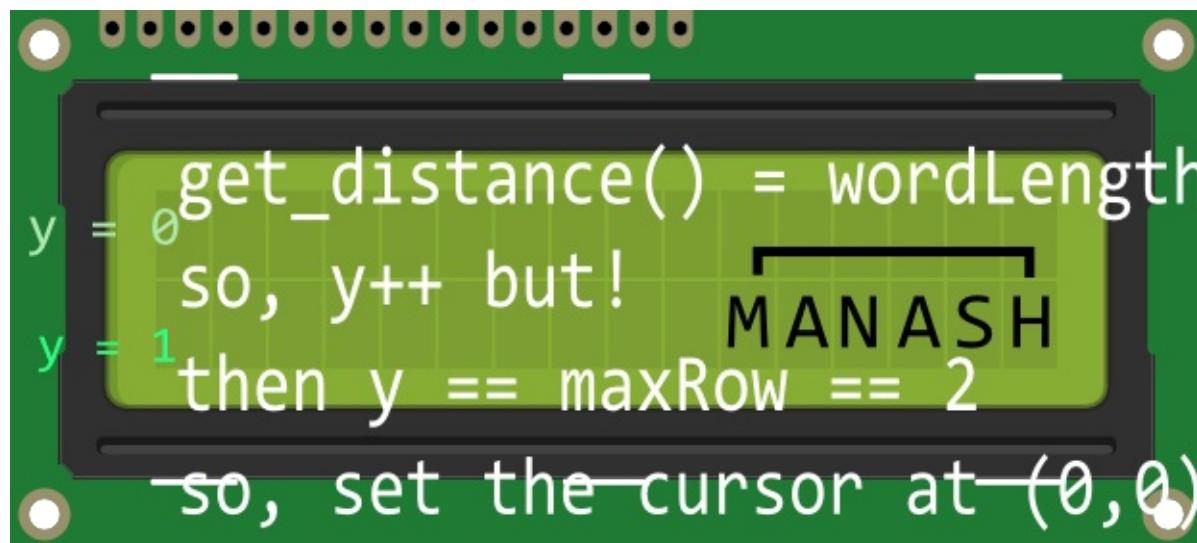
অ্যালগরিদম

চলবে:

- যদি স্ট্রিংয়ের প্রথম বর্ণথেকে LCD এর বাউন্ডাৰি (maxCol) এর দূৰত্ব শব্দের দৈর্ঘ্যের চেয়ে বেশি বা সমান হলে
হয়:
 - যে স্ট্রিংকে ক্রল কৱাতে হবে সেটাকে (x,y) তে প্রেস কৱে প্ৰিণ্ট কৱব
 - del সময় অপেক্ষা কৱব
 - LCD ছিয়াৰ কৱব
 - x এৰ মান ১ বাড়াৰ
 - y এৰ দূৰত্ব শব্দেৰ দৈর্ঘ্যেৰ সমান হয়:
 - y এৰ মান ১ বাড়াৰ
 - x এ ০ বসাব
- যদি y এৰ মান LCD এৰ maxRow এৰ সংখ্যাৰ সমান হয়:
 - x এৰ মান ০ বসাব
 - y এৰ মান ০ বসাব

ছৰিতে:





কাজের ধারা:

যেভাবে কাজটি করব

MyLiquidCrystal.h ফাইলে MyLiquidCrystal ক্লাস তৈরি করে LiquidCrystal ইনহেরিট করা

১। Header ফাইল লেখার নিয়ম

- হেডার ফাইল লেখার জন্য যে নাম দিয়ে হেডার ফাইলটি তৈরি করতে চান, `#ifndef` প্রিপ্রসেসর ব্যবহার করে সে নামটি সবার প্রথমে লিখতে হবে। যেমন এখানে: `#ifndef MyLiquidCrystal_h`
- তারপরে সেটাকে `#define` প্রিপ্রসেসর দিয়ে ডিফাইন করতে হবে। এখানে: `#define MyLiquidCrystal_h`
- সবার শেষে `#endif` ব্যবহার করতে হবে

একনজরে একটি হেডার ফাইল

```
//Header File Name: MyLiquidCrystal.h
#ifndef MyLiquidCrystal_h
#define MyLiquidCrystal_h

//Logic Goes here

#endif
```

২। প্রয়োজনীয় ফাইল #include করা

যেহেতু আমি `LiquidCrystal` লাইব্রেরিটি ও `Arduino.h` ব্যবহার করছি তাহলে সেটা `#include` করতে হবে।

সুতোঁ,

```
#ifndef MyLiquidCrystal_h
#define MyLiquidCrystal_h

//Naming unsigned 8 bit integer as "byte"
typedef uint8_t byte;

//Adding Necessary Header Files
#include "Arduino.h"
#include "LiquidCrystal.h"

#endif
```

৩। প্রযোজনীয় স্লাস, প্রোপার্টি ও মেথড তৈরি:

i) Inheritance প্রয়োগ

যেহেতু আমরা একটি স্লাস তৈরি করে LiquidCrystal কে ইনহেরিট করব সেকারণে Header ফাইলের নাম দিয়ে একটি স্লাস তৈরি করে কোলন : চিহ্ন দিয়ে আমরা LiquidCrystal স্লাসটিকে ইনহেরিট করব যাতে LiquidCrystal এর সকল মেথড, প্রোপার্টি আমরা ব্যবহার করতে পারি।

```
#ifndef MyLiquidCrystal_h
#define MyLiquidCrystal_h

//Naming unsigned 8 bit integer as "byte"
typedef uint8_t byte;

//Adding Necessary Header Files
#include "Arduino.h"
#include "LiquidCrystal.h"

//Added Class
class MyLiquidCrystal : public LiquidCrystal {
    //Methods And Properties go here
};

#endif
```

ii) প্রযোজনীয় Properties অ্যাড করা

আমার এই লাইব্রেরির জন্য কেশকিছু প্রোপার্টি অ্যাড করব। প্রোপার্টি ও মেথড প্রোগ্রামারভেদে পরিবর্তিত হয়। অনেকে সাধারণ কাজের জন্য অনেক মেথড ও প্রোপার্টি ব্যবহার করতে পারে (যেমন আমি) আর এক্সপেরিয়েন্স প্রোগ্রামারদের অনেক কম প্রোপার্টি ও মেথড হলেই চলে যায়। প্রোপার্টিগুলোকে সাধারণত private রাখা বেটার ও এই প্রোপার্টি ভ্যালু পরিবর্তন করার জন্য আমরা public get set মেথড ব্যবহার করতে পারি।

স্লাসের ডিতরের অংশ যদি বিবেচনায় আনি তাহলে,

```
class MyLiquidCrystal : public LiquidCrystal {
    private:
        //Initial Position Variable
        byte x;
        byte y;

        //Scroll Delay
        byte scrollDelay;

        //Storing LCD Row and Column
        byte lcdRows;
        byte lcdCols;

        //Getting the string length for out of bound calculation
        byte wordLength;

        //Distance from first character to the last boundary of lcd
        byte distance;
};
```

iii) প্রয়োজনীয় Method অ্যাড করা

আমার কাজের প্রয়োজনীয় প্রোপার্টি অ্যাড করে ফেললাম, এবার মেথড অ্যাড করা বাকি। মেথডের জন্য আমরা Access Modifier হিসেবে `public` ব্যবহার করব।

```

class MyLiquidCrystal : public LiquidCrystal {
public:

    //Function for scrolling
    void word_scroll(char *str, byte del);

    //Setting LCD Dimension
    void set_lcd(byte col, byte row);

    //Get LCD Dimension
    byte get_row(void);
    byte get_col(void);

    //Using Constructor from LiquidCrystal, and setting initial position to 0, 0 (or at t
MyLiquidCrystal(byte rs, byte en, byte d4, byte d5, byte d6, byte d7) : LiquidCrystal
{
    x = 0;
    y = 0;
}

//Getting the distance of word
byte get_distance(void);
};

```

দ্রষ্টব্য: MyLiquidCrystal এ আমরা LiquidCrystal এর কনস্ট্রাক্টর ব্যবহার করতে চাই। আর সুপার ক্লাসের কনস্ট্রাক্টর ব্যবহার করার ফরম্যাট হল এটা `derived_class(int arg1, int arg2) : base_class(arg1, arg2) {}`। এইভাবে কনস্ট্রাক্টর তৈরি করলে বেজ ক্লাসের কনস্ট্রাক্টর কল হয়। এই কাজটা করার কারণ সবার শেষে বলা হবে।

8 | সোর্স ফাইল লেখা [MyLiquidCrystal.cpp]

হেডার ফাইল তৈরি করার পরে সোর্স ফাইলে সেই ফাংশনগুলোর লজিক বসাতে হয়। এখানে আমরা যেসব মেথড ইম্প্লিমেন্ট করব, প্রত্যেকটি MyLiquidCrystal ক্লাসের। OOP এর স্ট্রাকচার অনুযায়ী, কোন ক্লাসের মেথড যদি Out of Class ইম্প্লিমেন্ট করতে হয় তাহলে তাকে এভাবে লিখতে হ্য: `return_type className :: (scope_resolution_operator) functionName(argument)`

ঠিক সেভাবে আমরা যদি সম্পূর্ণ কোড লিখি তাহলে,

```

#include "MyLiquidCrystal.h"

//Initialize LCD
void MyLiquidCrystal::set_lcd(byte col, byte row)
{
    lcdRows = row;
    lcdCols = col;
    begin(col, row);
}

```

```

}

//Getting Rows and Columns
byte MyLiquidCrystal::get_col(void)
{
    return lcdCols;
}

byte MyLiquidCrystal::get_row(void)
{
    return lcdRows;
}

//Getting Current Distance
//It is measured from the first character of the string upto the boundary

byte MyLiquidCrystal::get_distance(void)
{

    return get_col() + 1 - x;
}

//Function that actually does the scrolling
void MyLiquidCrystal::word_scroll(char *str, byte del)
{
    //Getting the string length
    wordLength = strlen(str);
    scrollDelay = del;

    //Getting row and column
    byte maxCol = get_col();
    byte maxRow = get_row();

    //Infinite Loop [You have to customize and solve it]
    //Buggy code
    while (true){
        if (get_distance() >= wordLength){

            /* Algorithm
            * -> Check if the distance is greater or equal to the wordlength
            * -> set Cursor at a coordinate
            * -> Print the string
            * -> wait about delay time
            * -> clear the lcd
            * -> Increase x one character
            * -> Check if the word is at boundary
            * -> if it's in the boundary then increase y and set x = 0
            * -> follow the algorithm from beginning
            * */
            setCursor(x, y);
            print(str);
            delay(scrollDelay);
            clear();
        }
    }
}

```

```

        x++;
        if (get_distance() == wordLength)
        {
            x = 0;
            y++;
        }
    }
    //If the cursor is at the bottom then setting the cursor at the beginning
    if(y == maxRow) {x = 0; y = 0;}
}
}

```

৫। Syntax Highlighting অ্যাড করার জন্য keywords.txt ফাইল এডিট করা

যদি আপনি চান Arduino IDE তে আপনার লাইব্রেরি অ্যাড করলে এটি অন্যসব লাইব্রেরির মত আপনার লাইব্রেরির ক্লাস, মেথড হাইলাইট করে দেবে তাহলে আপনার একটি ছোট্ট কাজ করতে হবে। লাইব্রেরির ডিরেষ্টরিতে keywords.txt ফাইল তৈরি করে সেখানে আপনার ক্লাস নেম লিখে ট্যাব (TAB) দিয়ে [অবশ্যই TAB দিতে হবে, স্পেস দিলে হবে না; ক্লাস/মেথড লিখে ট্যাব দিয়েছেন কিনা সেটা লক্ষ্য রাখলেই হবে, কয় স্পেসের ট্যাব হতে হবে সেটা গুরুত্বপূর্ণ বিষয় নয়]।

সাধারণত Class এর জন্য KEYWORD1 এবং Method এর জন্য KEYWORD2 ব্যবহার করা হয়। নিচে আমার তৈরি keywords.txt ফাইলটা দেওয়া হল।

```

#####
# Syntax Coloring Map For MyLiquidCrystal
#####

#####
# Datatypes (KEYWORD1)
#####

MyLiquidCrystal      KEYWORD1

#####
# Methods and Functions (KEYWORD2)
#####

word_scroll      KEYWORD2
set_lcd      KEYWORD2
get_row      KEYWORD2
get_col      KEYWORD2
get_distance     KEYWORD2

```

মোটামুটি প্রায় সবকাজ শেষ, বাকি থাকল শুধু একটা Example লেখা।

৬ | examples অ্যাড করা

আপনার লাইব্রেরি কীভাবে ব্যবহার করতে হয় সেটা যদি একটু দেখিয়ে দেন তাহলে যারা আপনার লাইব্রেরি ব্যবহার করবে তাদের জন্য সুবিধাজনক। examples অ্যাড করার জন্য আপনার লাইব্রেরির ডিরেক্টরিতে examples ফোল্ডারের ভিতর একটি ফোল্ডার তৈরি করুন। আমার ক্ষেত্রে ScrollText।

এই ফোল্ডারের ভিতরে হবহ এই ফোল্ডারের নাম দিয়ে একটা .ino ফাইল তৈরি করুন। আমি ScrollText.ino নামক ফাইল তৈরি করলাম।

ScrollText.ino ফাইলের ভিতরে আমি MyLiquidCrystal লাইব্রেরি ব্যবহার করে দেখাব।

```
#include <MyLiquidCrystal.h>

//Creating MyLiquidCrystal Object, constructor same as LiquidCrystal
MyLiquidCrystal myLcd(2, 3, 4, 5, 6, 7);

void setup()
{
    //Initializing the lcd module, set_lcd calls begin method so don't need to call lcd.begin()
    myLcd.set_lcd(20, 4);

    //String scroll method
    myLcd.word_scroll("Manash", 100);
}

void loop() {}
```

এভাবে আপনার একটি পরিপূর্ণ লাইব্রেরি তৈরি হয়ে গেল। এবাব যদি লাইব্রেরিটি ব্যবহার বা ডিবাগ করতে চান তাহলে নিচের কাজগুলো করতে হব।

তৈরিকৃত লাইব্রেরি ব্যবহার ও ডিবাগিং

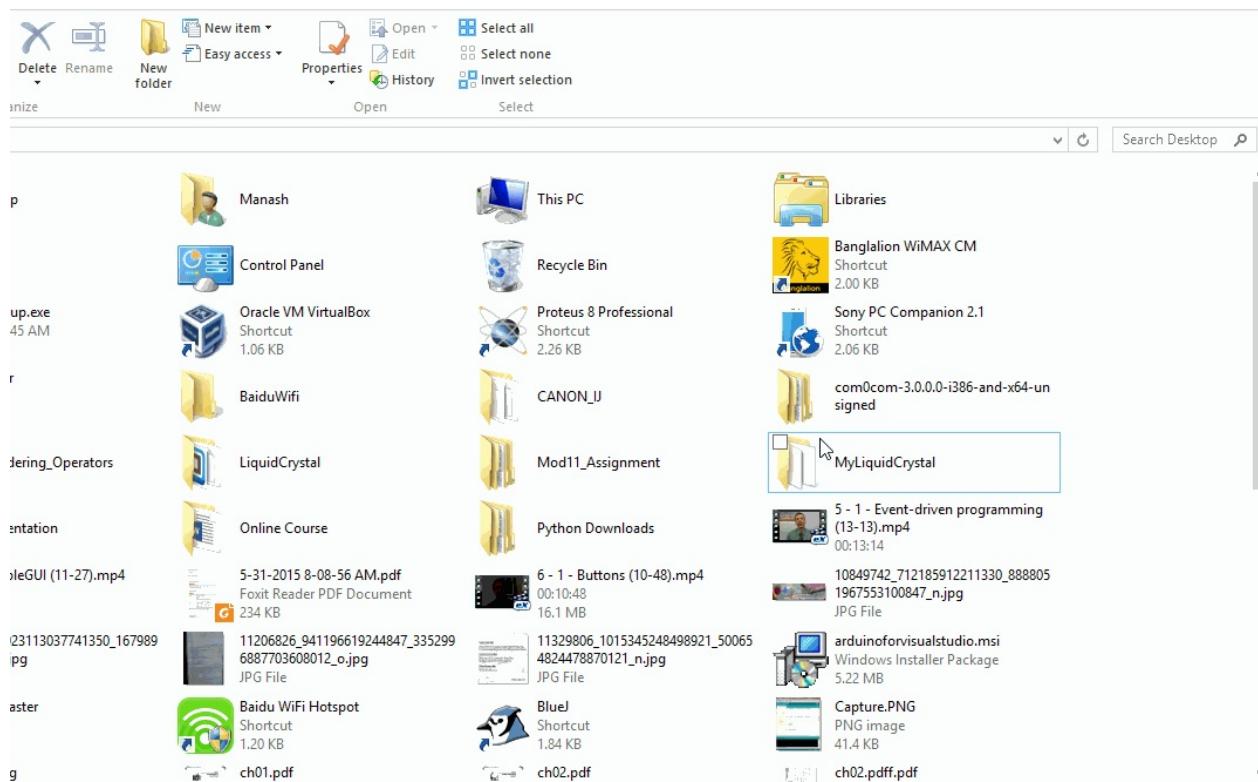
আপনার তৈরি করা লাইব্রেরি Arduino IDE তে দুইভাবে অ্যাড করে ব্যবহার করতে পারেন।

প্রথম পদ্ধতি: সরাসরি লাইব্রেরি Arduino IDE তে অ্যাড করে

তৈরি করা লাইব্রেরি ফোল্ডারটি কপি করুন ও C:\Users\user_name\Documents\Arduino\libraries\ এই ডিরেক্টরিতে পেস্ট করুন।

দ্বিতীয় পদ্ধতি: কম্প্রেসড (.zip) লাইব্রেরি অ্যাড করে

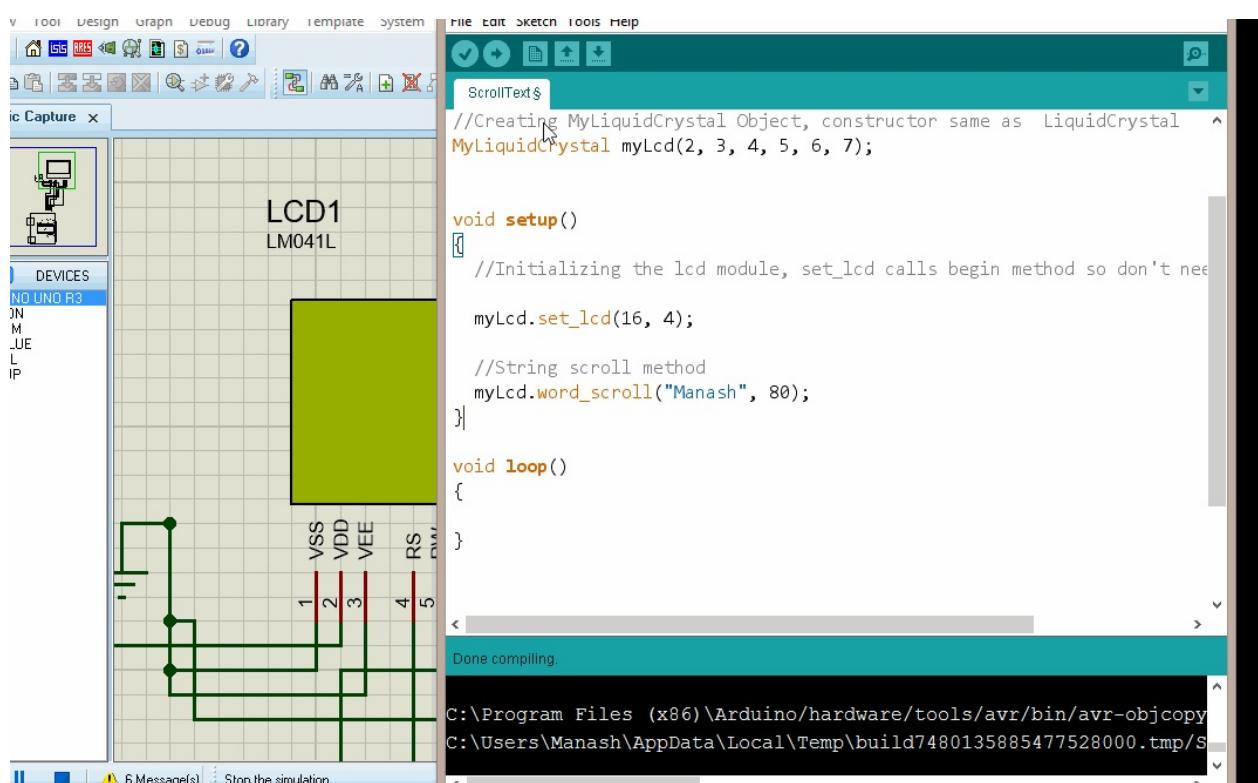
লাইব্রেরিটি জিপ করুন। তারপর Arduino IDE ওপেন করে Sketch > Import Library > your_library.zip সিলেক্ট করুন।



হাইলাইটিং না দেখতে পেলে Arduino IDE রিস্টার্ট করে দেখুন হাইলাইট করছে কিনা। তাও না করলে keywords.txt ফাইল চেক করে দেখুন ক্লাস, প্রোপার্টি বা মেথডের পরে TAB দিয়েছেন নাকি Space দিয়েছেন। Space দিলে হাইলাইটিং কাজ করবে না।

লাইব্রেরি ইন অ্যাকশন:

আপনি ইচ্ছা করলে আগের পর্বের প্রোটিয়াস ফাইলটা ব্যবহার করতে পারেন, কানেকশনের পরিবর্তন আনা হয় নি। আমি এখানে সিম্যুলেশনের বদলে বাস্তবে করে দেখালাম। এখানে আমি ScrollText.ino ফাইলটা আড়তইনোতে আঞ্চোড করেছি।



বাড়ির কাজ:

আমার তৈরি করা লাইব্রেরিতে কয়েকটি সমস্যা আছে। তার মধ্যে দুইটি স্লিপ করতে হবে,

১। word_scroll মেথডটিতে আমি while(true) বা একটি Infinite loop ব্যবহার করেছি। আপনাকে এমনভাবে লাইব্রেরিকে মডিফাই করতে হবে যেন আমি word_scroll মেথড টা কন্ট্রোল করতে পারি। অর্থাৎ, infinite loop ব্যবহার না করে স্ক্রলিং কন্ট্রোল করতে হবে। আপনি ইচ্ছা করলে Button, Serial ইত্যাদি সবকিছু ব্যবহার করে এটা করতে পারেন। মানে আপনার সল্যুশনটা সফটওয়্যার বা হার্ডওয়্যার যেকোন ভিত্তিক হতে পারে।

২। একটা জিনিস খেয়াল করেছেন, word_scroll মেথডটা তখনই কাজ করবে যখন শব্দের দৈর্ঘ্য LCD এর কলামের দৈর্ঘ্যের চেয়ে ছোট। আপনাকে এটাকে এমনভাবে মডিফাই করতে হবে যেন এই ফাংশনটি সব আকারের স্ট্রিং বা শব্দের জন্য কাজ করে।

কোড ও সিম্যুলেশন প্রজেক্ট ফাইল

প্রজেক্ট কোড ও সিম্যুলেশনের ফাইলগুলো ডাউনলোড করুন [এখান](#) থেকে।

অ্যাডভান্সড

Qt ও Arduino বেসিক

যা যা প্রয়োজন:

- Qt 4 বা 5 [ডাউনলোড ও ইন্সটলেশন টিউটোরিয়ালের জন্য দেখুন [এখানে](#)]
- C++ প্রোগ্রামিং ল্যাঙ্গুয়েজ সম্পর্কে ধারণা (বেসিক অবজেক্ট ও রিয়েন্টেড প্রোগ্রামিং জ্ঞানসহ, যদি কনসেপ্টগুলো ঝালাই করতে চান তাহলে ডিসিট করুন [এখানে](#))
- Arduino Board (সিম্যুলেশন করার জন্য অতিরিক্ত কিছু কাজ করতে হবে তাই আপাতত একটি বোর্ড থাকলে কাজ করতে সুবিধা হবে)
- Usb A to B Cable
- Arduino IDE

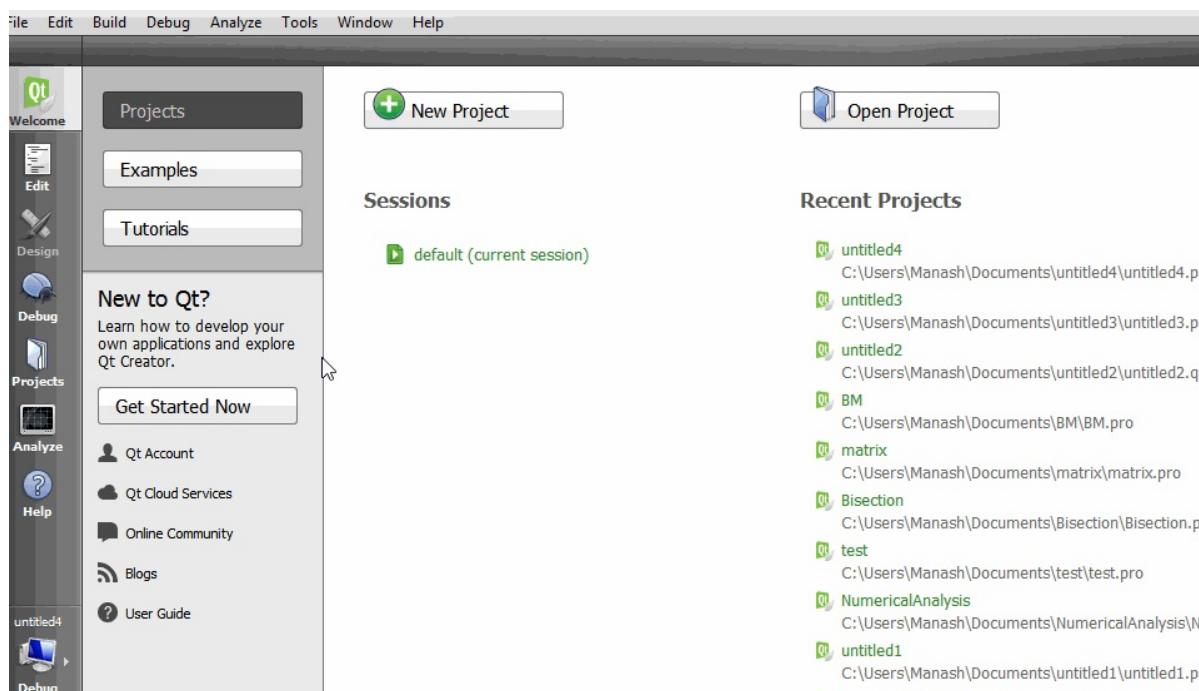
Qt কী?

Qt হল (উচ্চারণ হবে "কিউট" বা "কিউ-টি" (আনঅফিশিয়াল)) ক্রস প্ল্যাটফর্ম অ্যাপ্লিকেশন ফ্রেমওয়ার্ক যেটা দিয়ে যেকোন সফটওয়্যার বা হার্ডওয়্যার প্ল্যাটফর্মে সেম কোড বা অল্প পরিবর্তিত কোড দিয়েই সফটওয়্যার ডেভেলপ করা যায় যেগুলোর স্পিদ নেটিভ অ্যাপের সমান।

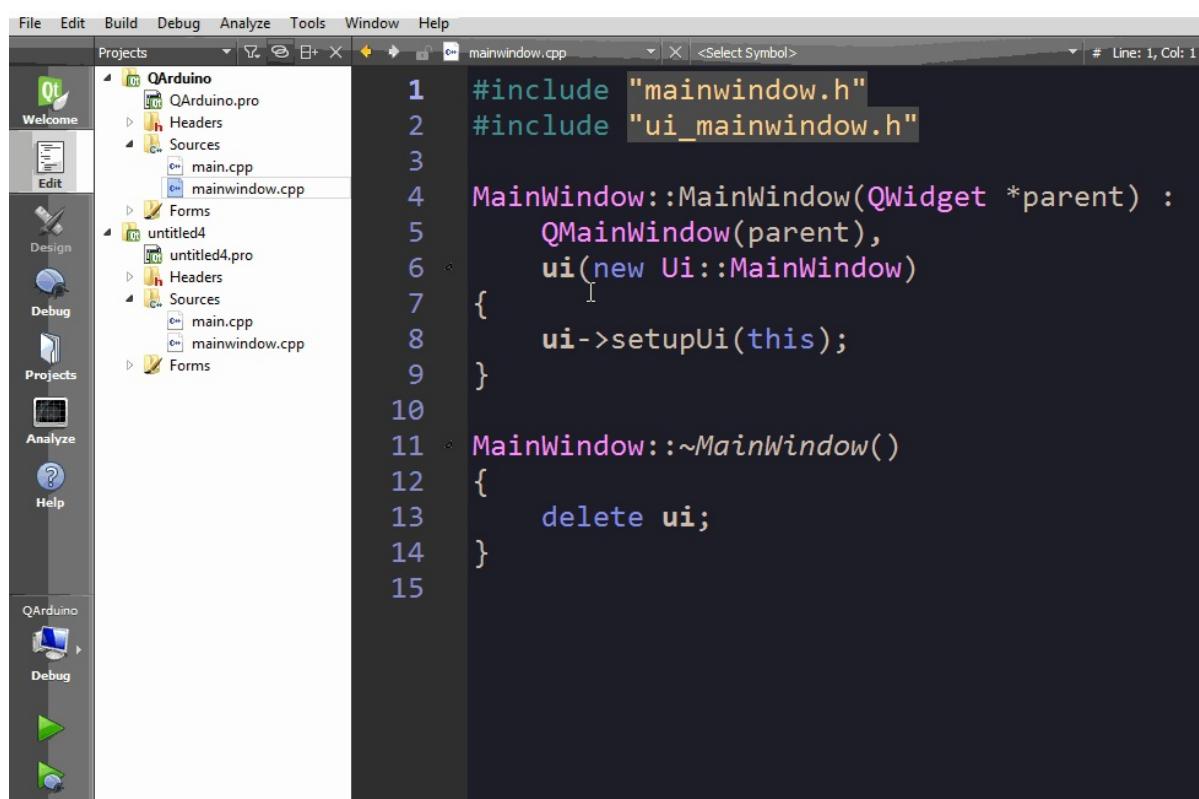
C++ এ GUI Based প্রোগ্রামিং টা বেশ ঝামেলাকর বলে মনে হলেও Qt Framework যারা ব্যবহার করেন তাদের ক্ষেত্রে গুই প্রোগ্রামিং খুবই সহজ। কয়েক লাইন লিখেই চমৎকার সব অ্যাপ ডেভেলপ করা যায়। অন্যান্য ফ্রেমওয়ার্কের সাথে এর বেশ কিছু মিল ও অনিল আছে। Qt Framework এর ইউনিক ফিচারের মধ্যে একটি signal ও slot। এছাড়াও অনেক ফিচার আছে কিন্তু যেহেতু এটা মূলত Arduino বেজড পোস্ট তাই কিউট নিয়ে একটু কম ই বলা হবে।

Qt First Run

- কিউট ফ্রেমওয়ার্ক ওপেন করুন -> New Project -> Application -> Qt Widgets Application -> Choose -> Name -> Next -> Next -> Next -> Finish দিলে আপনাকে MainWindow.cpp ফাইল নিয়ে আসবে



- এবার বামের ফাইল ডিরেক্টরি থেকে main.cpp সিলেক্ট করুন ডাবল ক্লিক করে তারপর নিচে-বামে-কোণায় দেখুন প্রে আইকন আছে সবুজ রংয়ের, সেটাতে ক্লিক করুন অথবা Ctrl+R চাপুন তাহলে প্রোগ্রামটি রান হবে ও আপনি একটি MainWindow দেখতে পাবেন



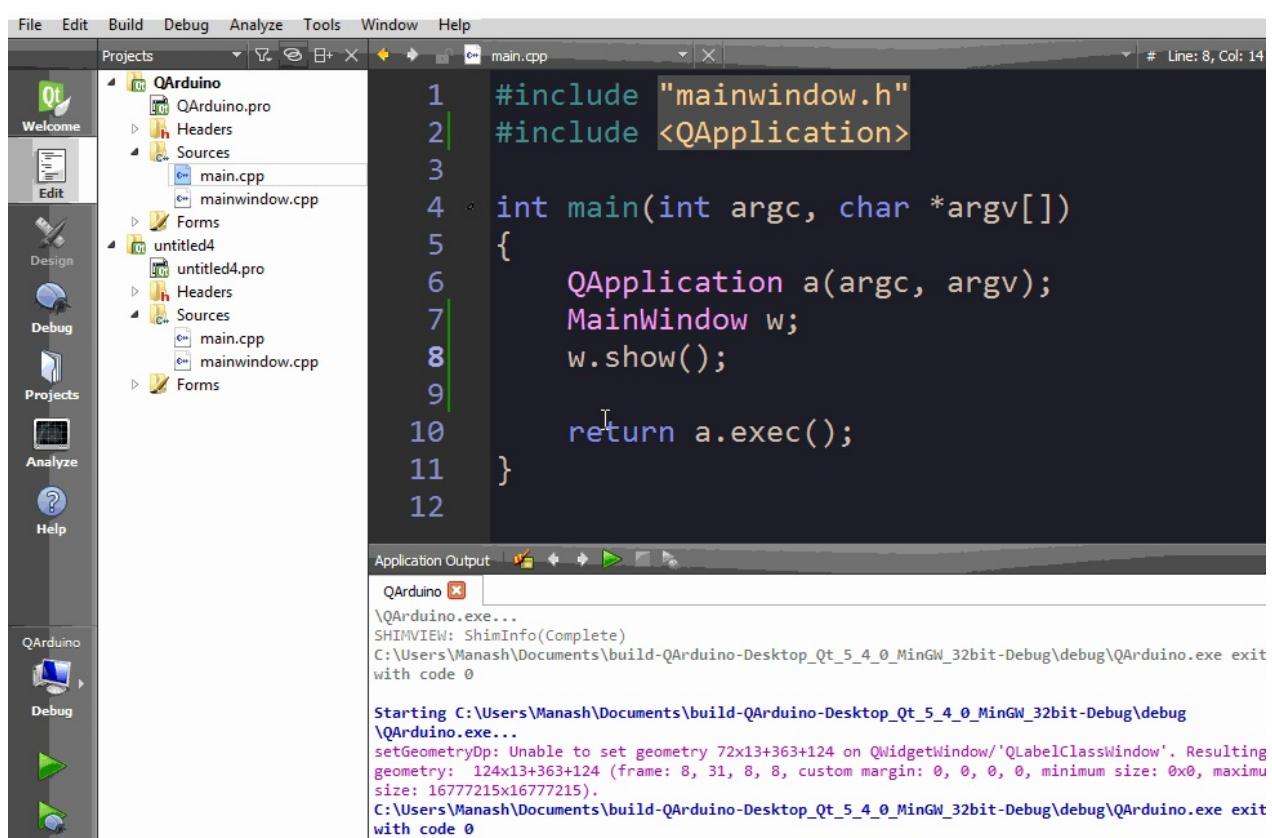
চালু হলে মেইনটেইন্ডো ক্লোজ করে দিন। এতে বুঝা গেল আপনার Qt ঠিকঠাক ইন্সটলড হয়েছে এবং আপনি কাজ করার জন্য প্রস্তুত।

Qt প্রথম প্রোগ্রাম

- আগের প্রজেক্ট থেকে main.cpp ফাইলটি সিলেক্ট করুন ও নিচের কোডটি লিখে রান করুন

```
#include "mainwindow.h"
#include <QApplication>
#include <QLabel>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    QLabel label("Electroscholars");
    label.show();
    return a.exec();
}
```



প্রোগ্রামটিতে আমরা কী করলাম? সাধারণত নতুন কোন ল্যাঙ্গুয়েজ শুরু করার সময় আমরা "Hello World" জাতীয় কিছু প্রিন্ট করি। Minimum Qt GUI অ্যাপ ডেভেলপ করার জন্যও আমরা সিম্পল GUI বেজড একটা টেক্সট শো করলাম। হ্যাঁ, লেবেলে "Electroscholars" রিপ্রেস করে আপনার নামও বসিয়ে দেখতে পাবেন।

Walkthru

```
#include "mainwindow.h"
```

প্রজেক্ট ক্রিয়েট করার সময় এই ফাইলটা main.cpp ফাইলে আইডিই অটো অ্যাড করে দিয়েছে। আমরা এই ফাইলে আপাতত কিছু লিখিনি তাই অ্যাড করলে বা না করলে কিছু যায় আসে না। আপনি ইচ্ছা করলে ডিলিট করে দিতে পারেন।

#include <QApplication>

Qt এ গুই প্রোগ্রামিং করতে হলে এই Header টা অবশ্যই অ্যাড করতে হবে। কারণ নিচে বলা হল।

#include <QLabel>

Qt এর ক্ষেত্রে দুইটা জিনিস খেয়াল করে দখবেন, Header এর নাম আর Class এর নাম একই হয়। আর সব কম্পোনেন্টগুলোর আগে Q থাকে। যেহেতু এইখানে Label কম্পোনেন্টটা নিয়ে কাজ করা হচ্ছে তাই আমরা Label এর Qt Variant QLabel ক্লাসটি ব্যবহার করার জন্য Header টি অ্যাড করেছি।

QApplication a(argc, argv);

শুরুতেই আমরা একটি চিপিক্যাল মেইন ফাংশন দেখলাম। এই মেইন ফাংশনের ভিতরে দেখা যাচ্ছে QApplication এর একটি অবজেক্ট a ক্রিয়েট করে কনস্ট্যুটর হিসেবে আমরা কমান্ড লাইন আর্গুমেন্ট গুলো পাস করছি (argc, argv)। QApplication অবজেক্ট ছাড়া কোন গুই প্রোগ্রাম চলবে না। কারণ অন্যসবকিছুর পাশাপাশি QApplication একটা Event Loop তৈরি করে। এই ইভেন্ট লুপের কাজ হল অ্যাপ্লিকেশন ততক্ষণ পর্যন্ত রান করা যতক্ষণ পর্যন্ত আপনি window এর জ্ঞাজ বাটনটি না চাপছেন।

QLabel label("....");

এখানে আমরা QLabel একটা অবজেক্ট তৈরি করেছি এবং কনস্ট্যুটর হিসেবে লেবেলে কী দেখাবে সেটি আর্গুমেন্ট হিসেবে দিয়েছি। এই অবজেক্টটি ক্রিয়েট করার সময় Invisible থাকে। তাই একে Visible করার জন্য আমরা label.show() মেথড টি ব্যবহার করেছি।

return a.exec()

এই মেথডটা মূলত Event Loop শুরু করে যেটা অ্যাপ্লিকেশনের Event গুলো অ্যাপ্রোপ্রিয়েট অবজেক্টগুলোর কাছে ফরওয়ার্ড করে। এই সব ইভেন্টকে User Action এর মাধ্যমে তৈরি হয় যেমন বাটনে ক্লিক করা। আমরা বাটনে ক্লিক করে কীভাবে আড্রিনো LED অফ ও অন করা যায় সেটা এই পর্বে দেখব। Event Loop তখন টার্মিনেট হয় যখন QApplication অবজেক্টের quit ফাংশনটি কল করা হয়।

বলাই বাহ্ল্য কিউটের অনেক কিছু এখানে স্কিপ করা হয়েছে। কিউট শেখার জন্য কিউট ডকুমেন্টেশন এর চেয়ে ভাল কিছু হতে পারে না। প্রজেক্ট বেজড কাজ শেখার জন্য Qt এর উপর নিচের বই ও ইউটিউব চ্যানেলগুলো দেখতে পারেন।

বইয়ের ও ডকুমেন্টেশনের তালিকা

- The Book of Qt 4 - The Art of Building Qt Application
- C++ GUI Programming in Qt - Summerfield
- Official Qt Documentation

ইউটিউব লিঙ্ক

- VoidRealms এর ১০০+ কিউট টিউটোরিয়াল
- Qt BootCamp

Qt Designer ব্যবহার করে সিম্পল Event Driven GUI অ্যাপ্লিকেশন

এখন আমরা এমন একটি অ্যাপ বানাব যার দুইটা বাটন থাকবে, একটি বাটনে ক্লিক করলে মেসেজ শো করবে Led On ও আরেকটি বাটনে ক্লিক করলে মেসেজ শো করবে Led Off। তাহলে শুরু করা যাবে।

- আগের প্রজেক্টটি ওপেন করা থাকলে সেটাকে এডিট করেই কাজ করা যাবে। চাইলে আপনি নতুন প্রজেক্ট তৈরি করে নিচের স্টেপ ফলো করতে পারেন। বামপাশের Forms ফোল্ডারের ডিতরে দেখুন mainwindow.ui ফাইলটি আছে, সেটাতে ডাবল ক্লিক করুন ও ইমেজ দেখে ঝটপট দুইটা বাটন ও একটি লেবেল বসিয়ে দিন।

```

2 #include < QApplication >
3 #include < QLabel >
4
5
6 int main(int argc, char *argv[])
7 {
8
9     QApplication a(argc, argv);
10
11     return a.exec();
12 }
13

```

Application Output

```

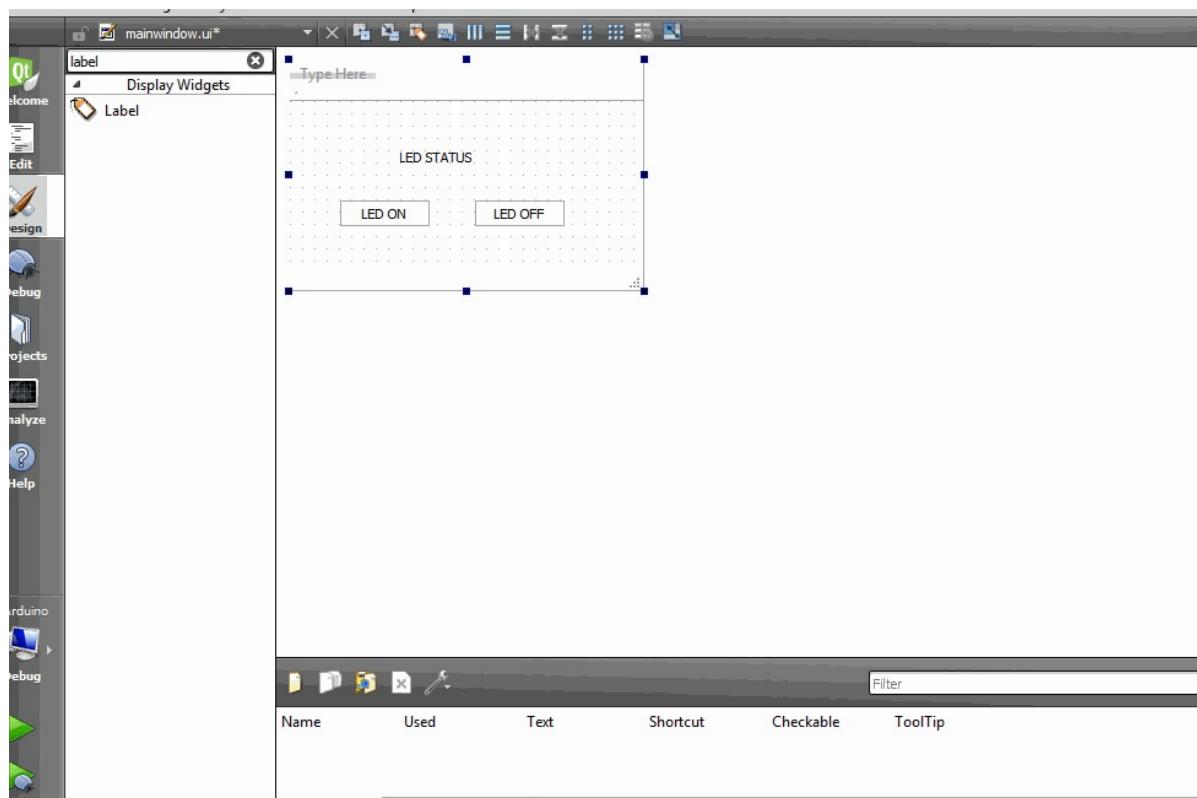
QArduino
SHIMVIEW: ShimInfo(Complete)
C:\Users\Manash\Documents\build-QArduino-Desktop_Qt_5_4_0_MinGW_32bit-Debug\debug\QArduino.exe exited normally after 0.000 seconds.

Starting C:\Users\Manash\Documents\build-QArduino-Desktop_Qt_5_4_0_MinGW_32bit-Debug\debug\QArduino.exe
setGeometryDp: Unable to set geometry 72x13+363+124 on QWidgetWindow/'QLabelClassWindow'. Resulting geometry: custom margin: 0, 0, 0, 0, minimum size: 0x0, maximum size: 16777215x16777215.
C:\Users\Manash\Documents\build-QArduino-Desktop_Qt_5_4_0_MinGW_32bit-Debug\debug\QArduino.exe exited normally after 0.000 seconds.

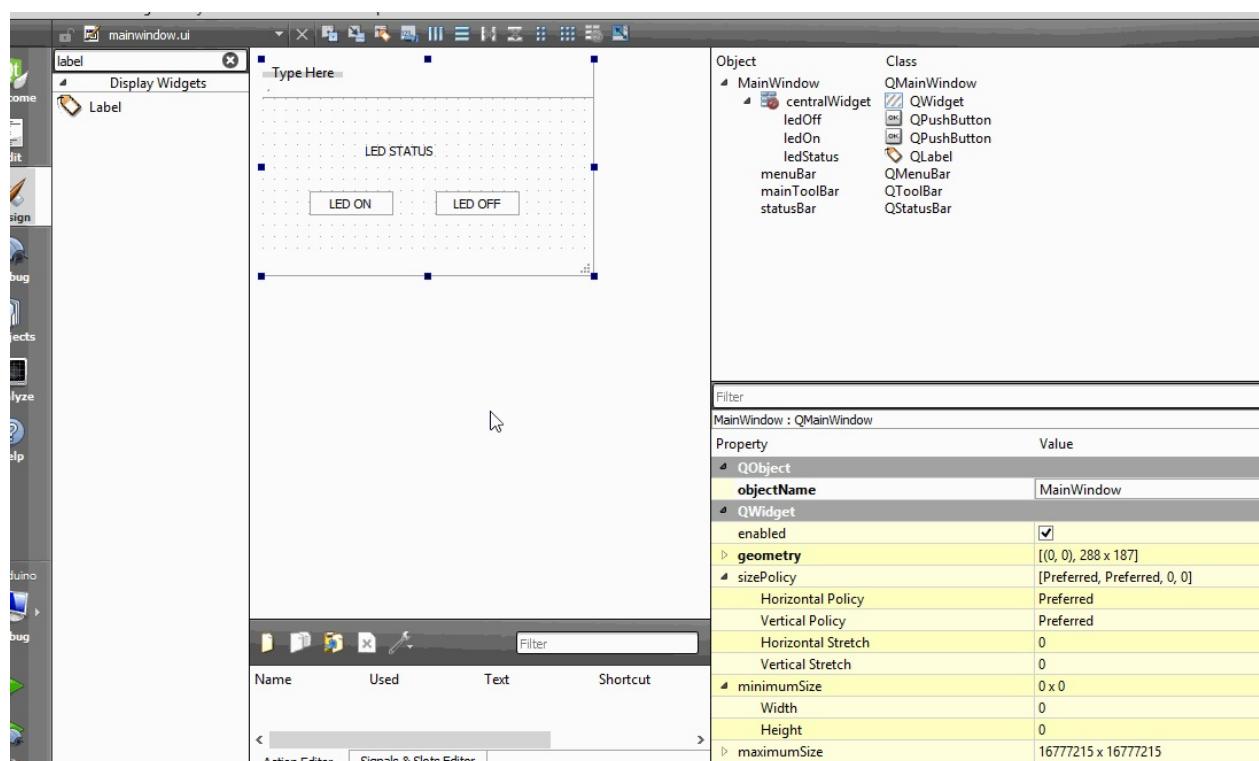
Starting C:\Users\Manash\Documents\build-QArduino-Desktop_Qt_5_4_0_MinGW_32bit-Debug\debug\QArduino.exe
setGeometryDp: Unable to set geometry 72x13+363+124 on QWidgetWindow/'QLabelClassWindow'. Resulting geometry: custom margin: 0, 0, 0, 0, minimum size: 0x0, maximum size: 16777215x16777215.
C:\Users\Manash\Documents\build-QArduino-Desktop_Qt_5_4_0_MinGW_32bit-Debug\debug\QArduino.exe exited normally after 0.000 seconds.

```

- দ্রষ্টব্য: বাটনের উপরের টেক্সট পরিবর্তন করা হয়েছে, অবজেক্টের নাম কিন্তু নয়। অবজেক্টের নাম পরিবর্তন করার জন্য এই কাজ করুন:



- এবার বাটনে ক্লিক করলে যাতে কাজ হয় আমরা সেই লজিক বসাব। সেজন্য যেটা করতে হবে, বাটনের উপর রাইট ক্লিক করুন ও go to slot -> clicked() এ গেলে IDE আপনাকে একটি কোড ব্লকে নিয়ে যাবে যেখানে বাটনে ক্লিক করলে কী হবে সে কোড আপনি বসাবেন



দ্রষ্টব্য: qDebug() কে কিউটের cout মনে করতে পারেন। কনসোলে আউটপুট দেখানোর কাজ করে এটা। যদি এর দেখায় তাহলে #include <QDebug> লাইনটা প্রোগ্রামের শুরুতে বসিয়ে নিন

- একই ভাবে led off বাটনের লজিক বসান

```

12
13     MainWindow::~MainWindow()
14     {
15         delete ui;
16     }
17
18     void MainWindow::on_ledOn_clicked()
19     {
20         qDebug() << "Led is on" << "\n";
21     }
22
23

```

Application Output

```

QArduino
Led is on
C:\Users\Manash\Documents\build-QArduino-Desktop_Qt_5_4_0_MinGW_32bit-Debug\debug\QArduino.exe exited with code 0

```

- এবার আমাদের কাজ হবে ledStatus নামক QLabel এ আমাদের ইচ্ছামত টেক্সট বসানো। মনে রাখবেন, গুই ডিজাইনারে আপনি অবজেক্ট ড্যাগ ড্রপ করলে অটোমেটিক ui_mainwindow.h ফাইলে সবকিছু অ্যাড হয়ে যায়। এগুলো সব করে দেয় Qt IDE তাই আমাদের অনেক কাজ ই করা লাগবে না। এই পোস্টটি অনেক কমপ্যাক্ট বিধায় অনেক এক্সপ্রেশন স্ট্রিপ করা হয়েছে, তবে যতটুকু না বুঝলেই নয় ততটাই বলা হয়েছে।
আরেকটু জানতে চাইলে গুগল, বই কিংবা ডকুমেন্টেশন ঘুঁটে দেখতে পারেন।
 - ledStatus নামের QLabel অবজেক্টের টেক্সট চেঞ্জ করার জন্য setText মেথডটি আছে। কিন্তু সমস্যা হল আপনি কীভাবে ledStatus অবজেক্টকে ধরে তার মেথড কল করবেন? আসলে আমরা যখন প্রজেক্ট তৈরি করি তখন অটোমেটিক MainWindow / Widget / Dialog কে Ui নেমস্পেসের ভিতরে নিয়ে আসে এবং private অ্যাক্সেস মডিফায়ারের ভিতরে *ui নামে একটি অবজেক্ট ক্রিয়েট করে। আপাতত এইটুকু জানুন, আপনি ডিজাইনার দিয়ে যত কম্পোনেন্ট অ্যাড করবেন সেগুলো ui এর মধ্যে চলে আসে। ফলে আমরা যদি কোন কম্পোনেন্টের মেথড কল করতে চাই তাহলে ui->object->objectMethod()
অর্থাৎ আমি যদি ledStatus এর টেক্সট পরিবর্তন করতে চাই তাহলে লিখব ui->ledStatus->setText("Some text")।

```

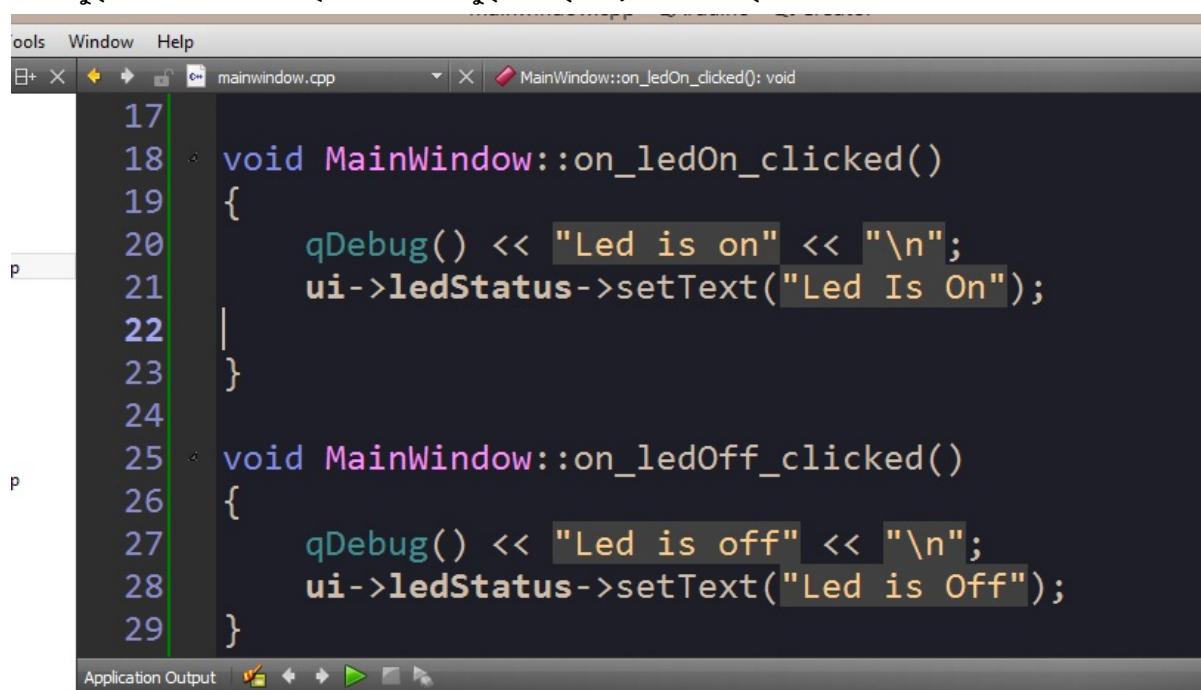
private slots:
    void on_ledOn_clicked();

    void on_ledOff_clicked();

private:
    Ui::MainWindow *ui;
};


```

- বাটন দুইটায় আমরা নিচের ছবির মত করে দুইটা লাইন অ্যাড করে দেই



```

17
18     void MainWindow::on_ledOn_clicked()
19     {
20         qDebug() << "Led is on" << "\n";
21         ui->ledStatus->setText("Led Is On");
22     }
23
24
25     void MainWindow::on_ledOff_clicked()
26     {
27         qDebug() << "Led is off" << "\n";
28         ui->ledStatus->setText("Led is Off");
29     }

```

- ফলাফল

The screenshot shows the Qt Creator IDE interface. On the left is the project tree under the 'Arduino' category, containing files like QArduino.pro, mainwindow.h, main.cpp, and mainwindow.cpp. The main window displays C++ code for a QMainWindow class. The code includes two slots: on_ledOn_clicked() and on_ledOff_clicked(). Each slot outputs a message to qDebug() and updates a QTextEdit control named 'ledStatus' with either 'Led Is On' or 'Led Is Off'. Below the code editor is the 'Application Output' tab, which shows the terminal output of the application. The output starts with 'Starting C:\Users\Manash\Documents\build-QArduino-Desktop_Qt_5_4_0_MinGW_32bit-Debug\debug\QArduino.exe...', followed by several lines of text: 'Led is on', 'Led is off', 'Led is on', 'Led is off', and 'Led is on'. The final line is 'C:\Users\Manash\Documents\build-QArduino-Desktop_Qt_5_4_0_MinGW_32bit-Debug\debug\QArduino.exe exited with code 0'.

```

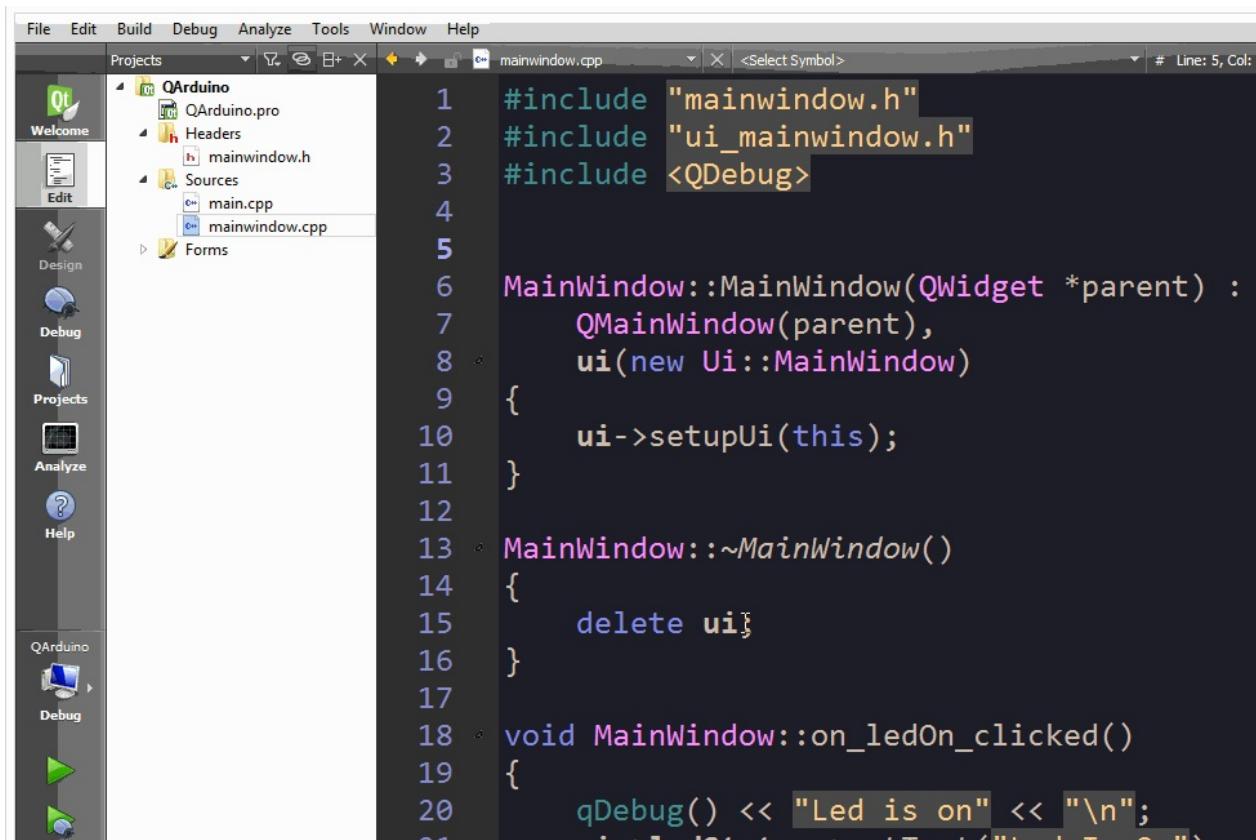
19 {
20     qDebug() << "Led is on" << "\n";
21     ui->ledStatus->setText("Led Is On");
22 }
23
24 void MainWindow::on_ledOff_clicked()
25 {
26     qDebug() << "Led is off" << "\n";
27     ui->ledStatus->setText("Led is Off");
28 }
29
30

```

তাহলে আমরা দেখলাম কীভাবে দুইটা বাটন ও লেবেল দিয়ে একটা গুই অ্যাপ তৈরি করা যায়। এবার আডুইনোর সাথে কমিউনিকেট করে আডুইনো led off ও on করতে পারে এমন একটি অ্যাপ আমরা তৈরি করব।

QSerialPort ও Arduino

- সিরিয়াল কম্যুনিকেশন এস্টেলিশ করার জন্য আমরা কিউট অফিশিয়াল QSerialPort লাইব্রেরিটা ব্যবহার করব। লাইব্রেরিটা ব্যবহার করার আগে আপনার প্রজেক্টের .pro ফাইলে serialport অ্যাড করতে হবে।



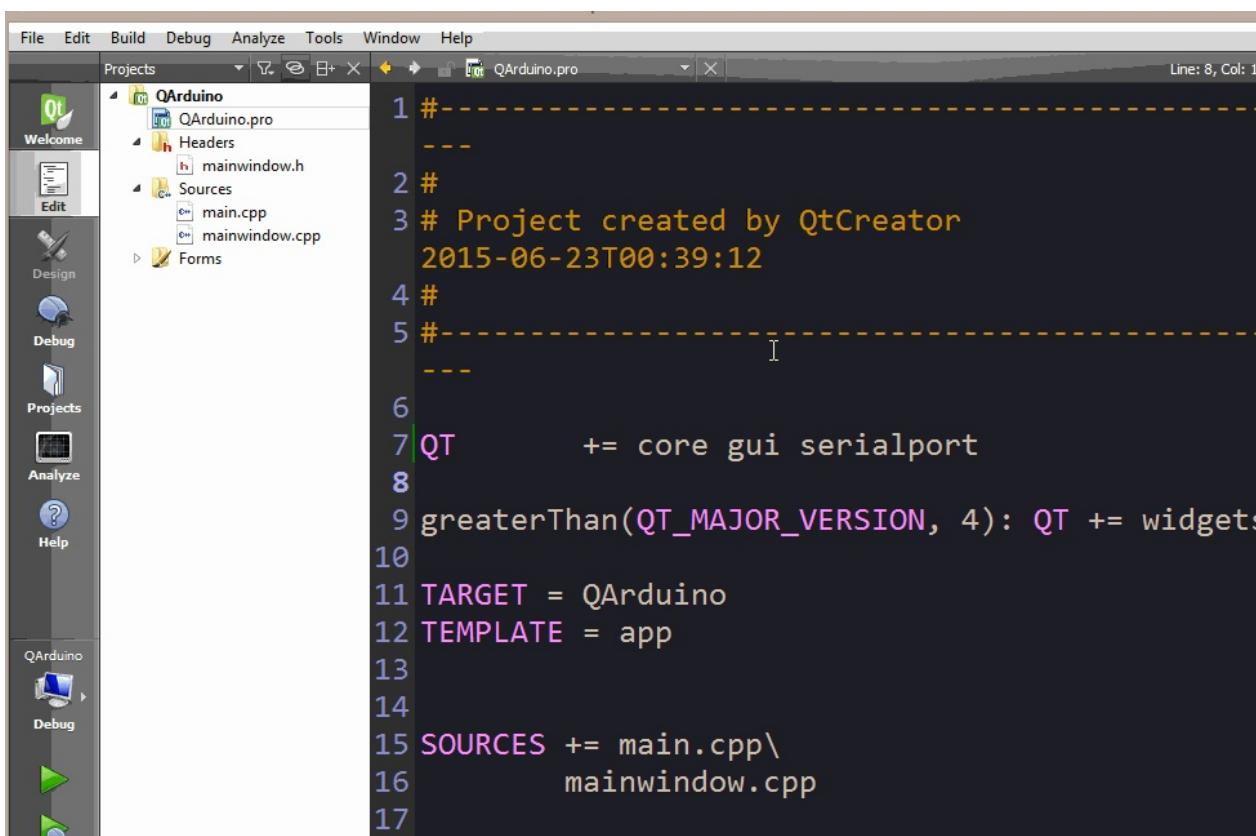
```

File Edit Build Debug Analyze Tools Window Help
Projects QArduino
  QArduino.pro
  Headers mainwindow.h
  Sources main.cpp mainwindow.cpp
  Forms

1 #include "mainwindow.h"
2 #include "ui_mainwindow.h"
3 #include <QDebug>
4
5
6 MainWindow::MainWindow(QWidget *parent) :
7     QMainWindow(parent),
8     ui(new Ui::MainWindow)
9 {
10     ui->setupUi(this);
11 }
12
13 MainWindow::~MainWindow()
14 {
15     delete ui;
16 }
17
18 void MainWindow::on_ledOn_clicked()
19 {
20     qDebug() << "Led is on" << "\n";
21 }

```

- এখন আপনি QSerialPort লাইব্রেরি ব্যবহার করার জন্য প্রস্তুত | mainwindow.h Header ফাইলটা ওপেন করুন `#include <QSerialPort>` সবার উপরে লিখুন Header ফাইলটা অ্যাড করার জন্য ও QSerialPort নামের একটি অবজেক্ট পয়েন্টার তৈরি করুন। এধরণের অবজেক্টে সাধারণত private access modifier ব্যবহার করা হয়। আপনি চাইলে public ও ব্যবহার করতে পারেন



```

File Edit Build Debug Analyze Tools Window Help
Projects QArduino
  QArduino.pro
  Headers mainwindow.h
  Sources main.cpp mainwindow.cpp
  Forms

1 #
2 #
3 # Project created by QtCreator
4 2015-06-23T00:39:12
5 #
6
7 QT      += core gui serialport
8
9 greaterThan(QT_MAJOR_VERSION, 4): QT += widgets
10
11 TARGET = QArduino
12 TEMPLATE = app
13
14
15 SOURCES += main.cpp \
16             mainwindow.cpp
17

```

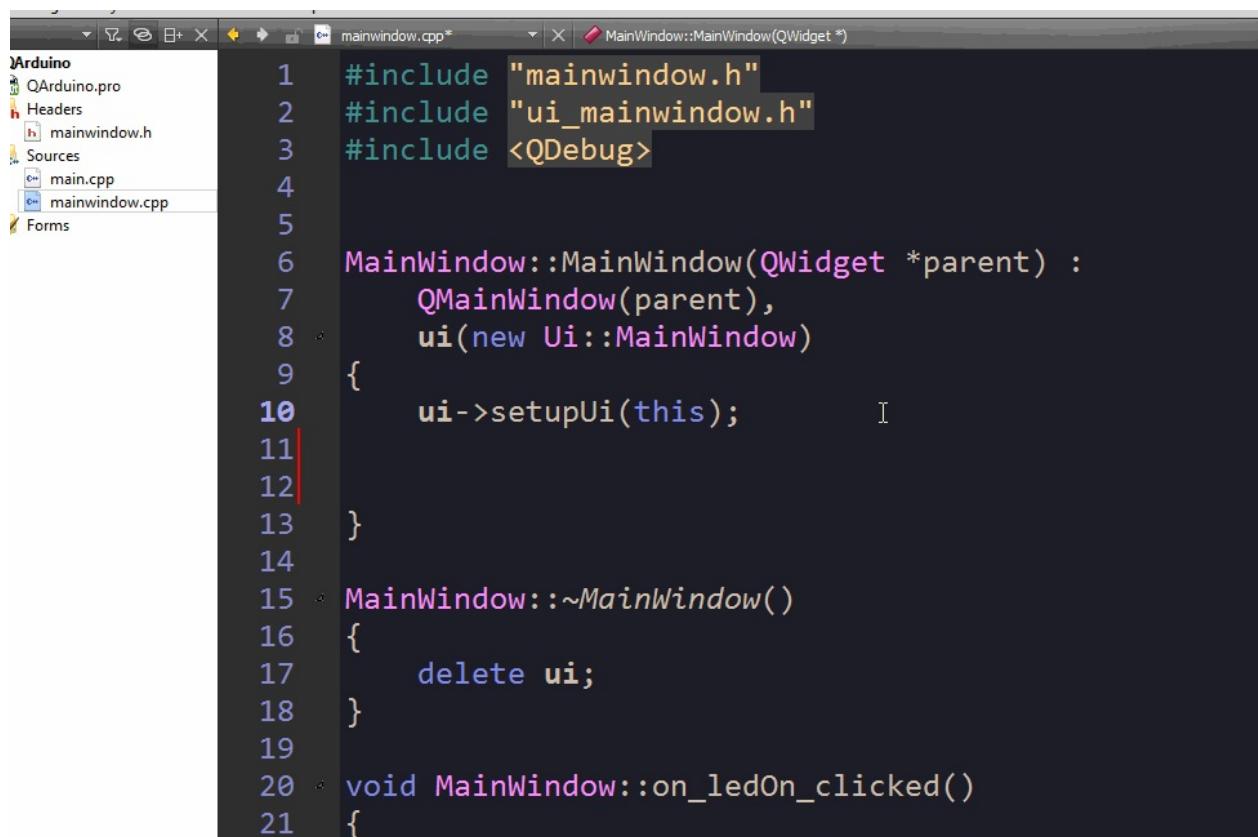
- গুরুত্বপূর্ণ: আমরা Arduino ও Qt কে ডিফল্ট কম্যুনিকেশন সেটিংসে কম্যুনিকেট করব। যদি আপনি আর্ডুইনোর অফিশিয়াল ওয়েবসাইটে সিরিয়াল চ্যাপ্টারটি দেখেন তাহলে দেখবেন বলা আছে An optional second argument configures the data, parity, and stop bits. The default is 8 data bits, no parity, one stop bit. অর্থাৎ, dataBits এর সংখ্যা 8 বা এক ফ্রেমে 8 বিট পাস হবে, কোন এক্সট্রা বা Parity বিট নাই এবং স্টপ বিট one stop। যেহেতু আমরা এখন ডিফল্ট কনফিগারেশন জানি তাহলে এই কনফিগারেশনে আমাদের কিউট অ্যাপটা ডেভেলপ করলেই হচ্ছে। যদি আপনি কনফিগ পরিবর্তন করতে চান সেক্ষেত্রে আপনার আর্ডুইনো কোডে Serial.begin মেথডে baudRate পাঠানোর পাশাপাশি কাস্টম কনফিগারেশন পাঠাতে হবে।
 - আমরা Qt এর সিরিয়ালপোর্টে এই প্রোপার্টিগুলো সেট করে দিতে পারি। যেহেতু তৈরি করা QSerialPort অবজেক্টের নাম দেওয়া হয়েছে arduino, তাই আমরা একটি বাটন তৈরি করব Connect নামে যাতে সেটাতে ক্লিক করলে Connection Established হ্য।

আর্ডুইনোর সাথে সিরিয়াল কানেকশন তৈরি করার জন্য (ডিফল্ট কনফিগারেশনে) প্রয়োজনীয় মেথড ও আর্গুমেন্ট:

```
arduino->setPortName("COM12"); //yours maybe different, please check by connecting it
arduino->setBaudRate(QSerialPort::Baud9600);
arduino->setDataBits(QSerialPort::Data8);
arduino->setParity(QSerialPort::NoParity);
arduino->setStopBits(QSerialPort::OneStop);
// Skipping hw/sw control
arduino->setFlowControl(QSerialPort::NoFlowControl);
```

- Connect বাটন বসানোর আগে কনস্ট্রাইক্টরে (mainwindow.cpp তে) আপনি QSerialPort অবজেক্টটি ইনিশিয়ালাইজ করতে পারেন নিচের কোডের মাধ্যমে। তারপর এর সিরিয়াল পোর্ট নেম বসিয়ে দিতে পারেন। আমার ক্ষেত্রে COM12।

```
arduino = new QSerialPort(this);
arduino->setPortName("COM12");
```

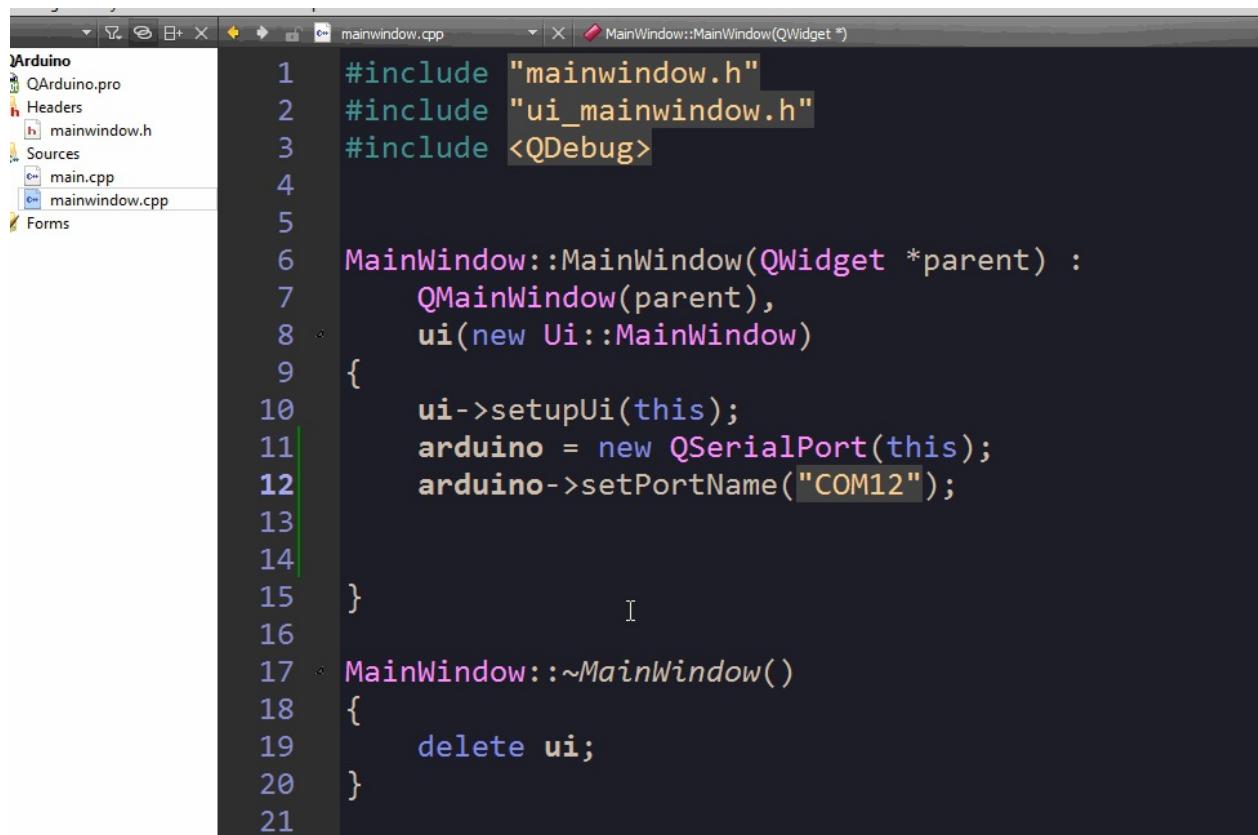


```

1 #include "mainwindow.h"
2 #include "ui_mainwindow.h"
3 #include <QDebug>
4
5
6 MainWindow::MainWindow(QWidget *parent) :
7     QMainWindow(parent),
8     ui(new Ui::MainWindow)
9 {
10     ui->setupUi(this);
11 }
12
13 }
14
15 MainWindow::~MainWindow()
16 {
17     delete ui;
18 }
19
20 void MainWindow::on_ledOn_clicked()
21 {

```

- দ্রষ্টব্য: পোর্ট দেখার জন্য

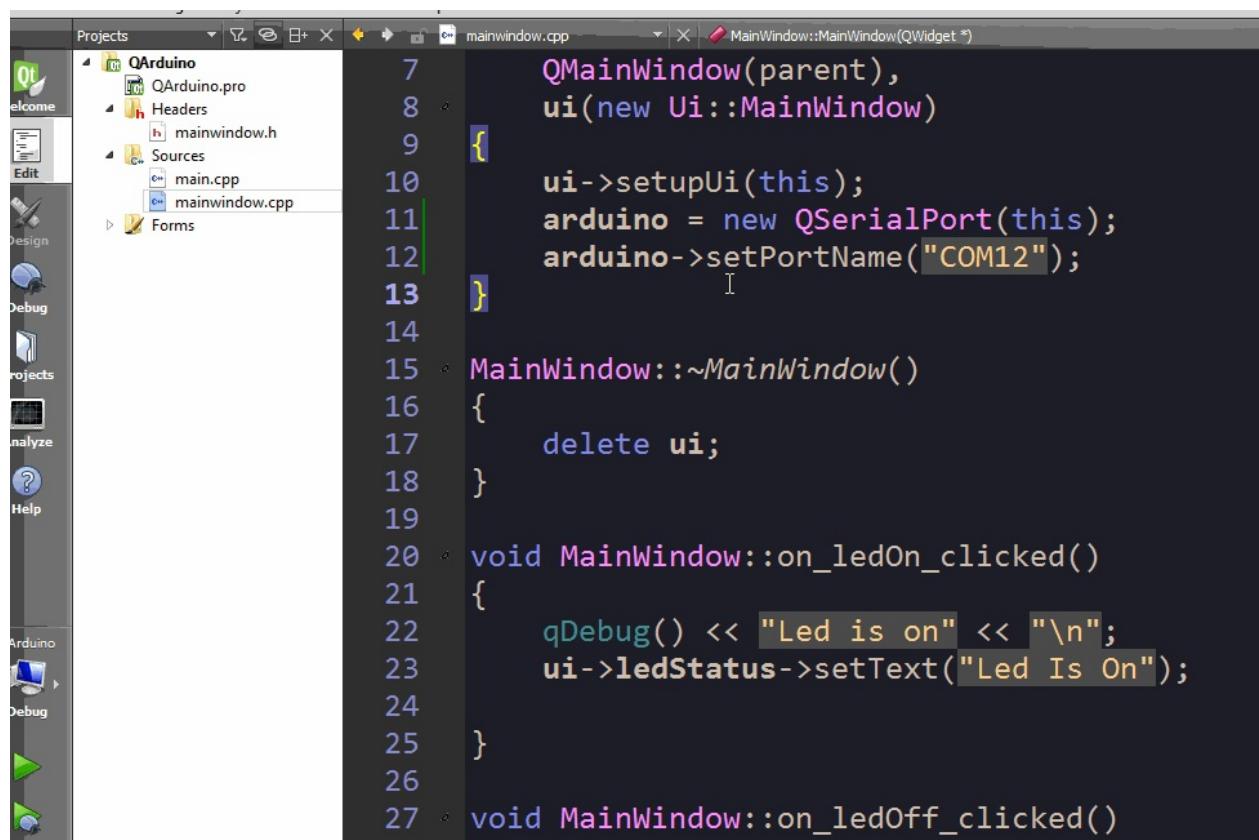


```

1 #include "mainwindow.h"
2 #include "ui_mainwindow.h"
3 #include <QDebug>
4
5
6 MainWindow::MainWindow(QWidget *parent) :
7     QMainWindow(parent),
8     ui(new Ui::MainWindow)
9 {
10     ui->setupUi(this);
11     arduino = new QSerialPort(this);
12     arduino->setPortName("COM12");
13
14 }
15
16 MainWindow::~MainWindow()
17 {
18     delete ui;
19 }
20
21

```

- এবার mainwindow.ui ফাইল সিলেক্ট করুন ও আরেকটি পুশ বাটন অ্যাড করে দিন। QPushButton এর ObjectName পরিবর্তন করে connectArduino বা যেকোন কিছু দিন আর উপরের Text পরিবর্তন করে Connect লিখে দিন তারপর রাইট ক্লিক করে go to slot দিন



```

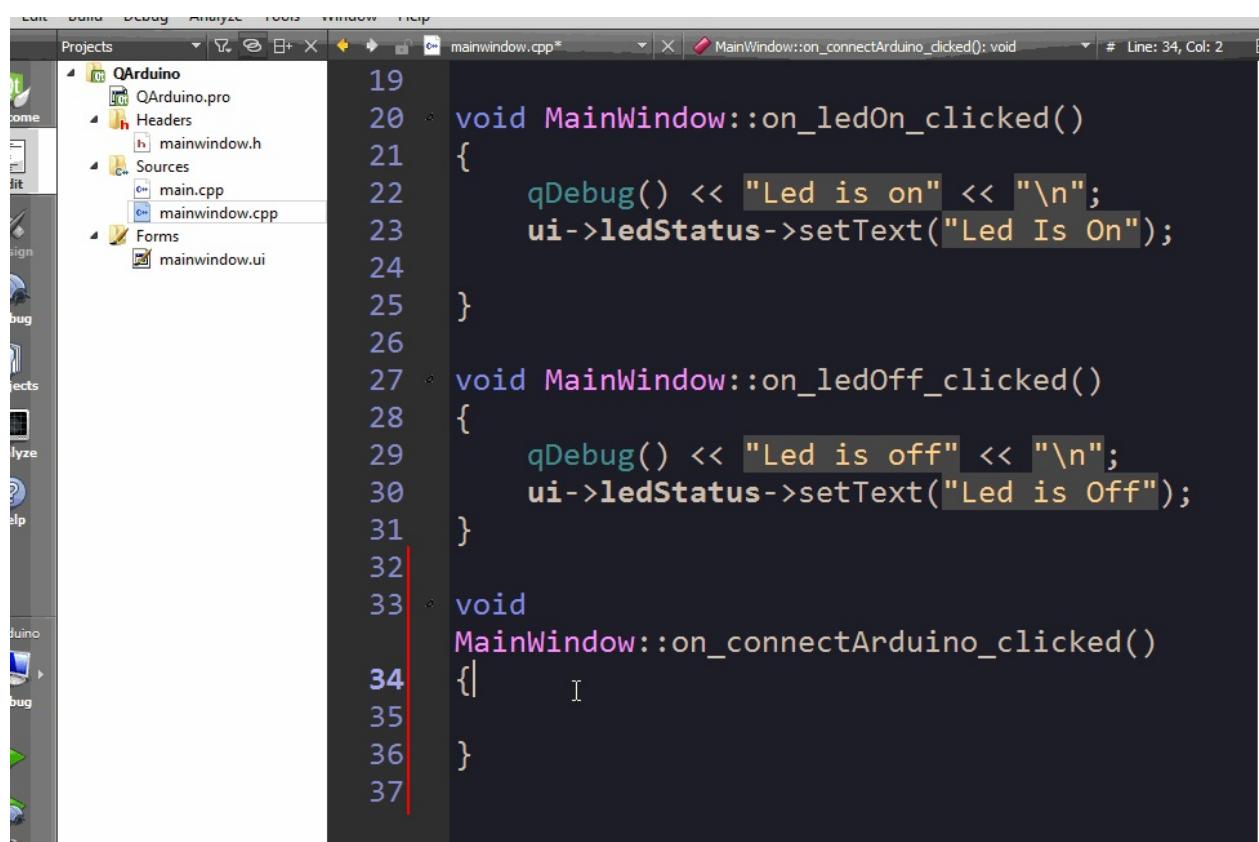
Projects QArduino
    QArduino.pro
    Headers
        mainwindow.h
    Sources
        main.cpp
        mainwindow.cpp
    Forms
        mainwindow.ui

mainwindow.cpp

7     QMainWindow(parent),
8         ui(new Ui::MainWindow)
9     {
10     ui->setupUi(this);
11     arduino = new QSerialPort(this);
12     arduino->setPortName("COM12");
13 }
14
15 MainWindow::~MainWindow()
16 {
17     delete ui;
18 }
19
20 void MainWindow::on_ledOn_clicked()
21 {
22     qDebug() << "Led is on" << "\n";
23     ui->ledStatus->setText("Led Is On");
24 }
25
26
27 void MainWindow::on_ledOff_clicked()

```

- বাটনটি ক্লিক করলে আমরা প্রথমে QSerialPort টা আগে জ্বাল করে দিব। কারণ যদি কোন কারণে ওপেন থাকে তাহলে সেটা কাজ করবে না। জ্বাল করে তারপর কানেকশন Establish করব। আর কানেক্ষ করার আগে দেখে নিব সে ReadWrite Mode এ ওপেন হচ্ছে কিনা। যদি ওপেন হয় তাহলে তার ডিফল্ট প্রোপার্টিগুলো সেট করে দিব:



```

Projects QArduino
    QArduino.pro
    Headers
        mainwindow.h
    Sources
        main.cpp
        mainwindow.cpp
    Forms
        mainwindow.ui

mainwindow.cpp

19
20 void MainWindow::on_ledOn_clicked()
21 {
22     qDebug() << "Led is on" << "\n";
23     ui->ledStatus->setText("Led Is On");
24 }
25
26
27 void MainWindow::on_ledOff_clicked()
28 {
29     qDebug() << "Led is off" << "\n";
30     ui->ledStatus->setText("Led is Off");
31 }
32
33 void
34 MainWindow::on_connectArduino_clicked()
35 {
36 }
37

```

- কানেক্ট করার পর মেসেজ শো করানোর জন্য আমরা QMessageBox ব্যবহার করতে পারি। এর মেথড ও প্রোপার্টি Qt Docs এ পাওয়া যাবে।

The screenshot shows the Qt Creator IDE. The main window displays the `mainwindow.cpp` file with the following code:

```

4
5
6
7 MainWindow::MainWindow(QWidget *parent) :
8     QMainWindow(parent),
9     ui(new Ui::MainWindow)
10 {
11     ui->setupUi(this);
12     arduino = new QSerialPort(this);
13     arduino->setPortName("COM12");
14 }
15
16 MainWindow::~MainWindow()

```

The application output window shows the following logs:

```

QArduino [x]
Led is on
Led is on
Led is on
C:\Users\Manash\Documents\build-QArduino-Desktop_Qt_5_4_0_MinGW_32bit-Debug\debug\QArduino.exe exited with code 0
Starting C:\Users\Manash\Documents\build-QArduino-Desktop_Qt_5_4_0_MinGW_32bit-Debug\debug\QArduino.exe...
Led is on
C:\Users\Manash\Documents\build-QArduino-Desktop_Qt_5_4_0_MinGW_32bit-Debug\debug\QArduino.exe exited with code 0

```

- প্রায় সব কাজ শেষ, এখন LED ON বাটন চাপ দিলে যাতে এমন কোন ডেটা পাস হয় যেটা আডুইনো চিহ্নিত করে Led জুলাতে পারে সেই কোডটি বসাব। LED OFF এর ক্ষেত্রে অন্য যেকোন ডেটার জন্য আডুইনো LED নিভিয়ে দিবে। Arduino তে ডেটা রাইট করার জন্য write() ফাংশনটি ব্যবহার করব

The screenshot shows the Qt Creator IDE. The main window displays the `mainwindow.cpp` file with the following code:

```

20
21 void MainWindow::on_ledOn_clicked()
22 {
23     qDebug() << "Led is on" << "\n";
24     ui->ledStatus->setText("Led Is On");
25 }
26
27
28 void MainWindow::on_ledOff_clicked()
29 {
30     qDebug() << "Led is off" << "\n";
31     ui->ledStatus->setText("Led is Off");
32 }

```

The application output window shows the following logs:

```

QArduino [x]
Led is on
C:\Users\Manash\Documents\build-QArduino-Desktop_Qt_5_4_0_MinGW_32bit-Debug\debug\QArduino.exe exited with code 0
Starting C:\Users\Manash\Documents\build-QArduino-Desktop_Qt_5_4_0_MinGW_32bit-Debug\debug\QArduino.exe...
Led is on
C:\Users\Manash\Documents\build-QArduino-Desktop_Qt_5_4_0_MinGW_32bit-Debug\debug\QArduino.exe exited with code 0
Starting C:\Users\Manash\Documents\build-QArduino-Desktop_Qt_5_4_0_MinGW_32bit-Debug\debug\QArduino.exe...
C:\Users\Manash\Documents\build-QArduino-Desktop_Qt_5_4_0_MinGW_32bit-Debug\debug\QArduino.exe exited with code 0

```

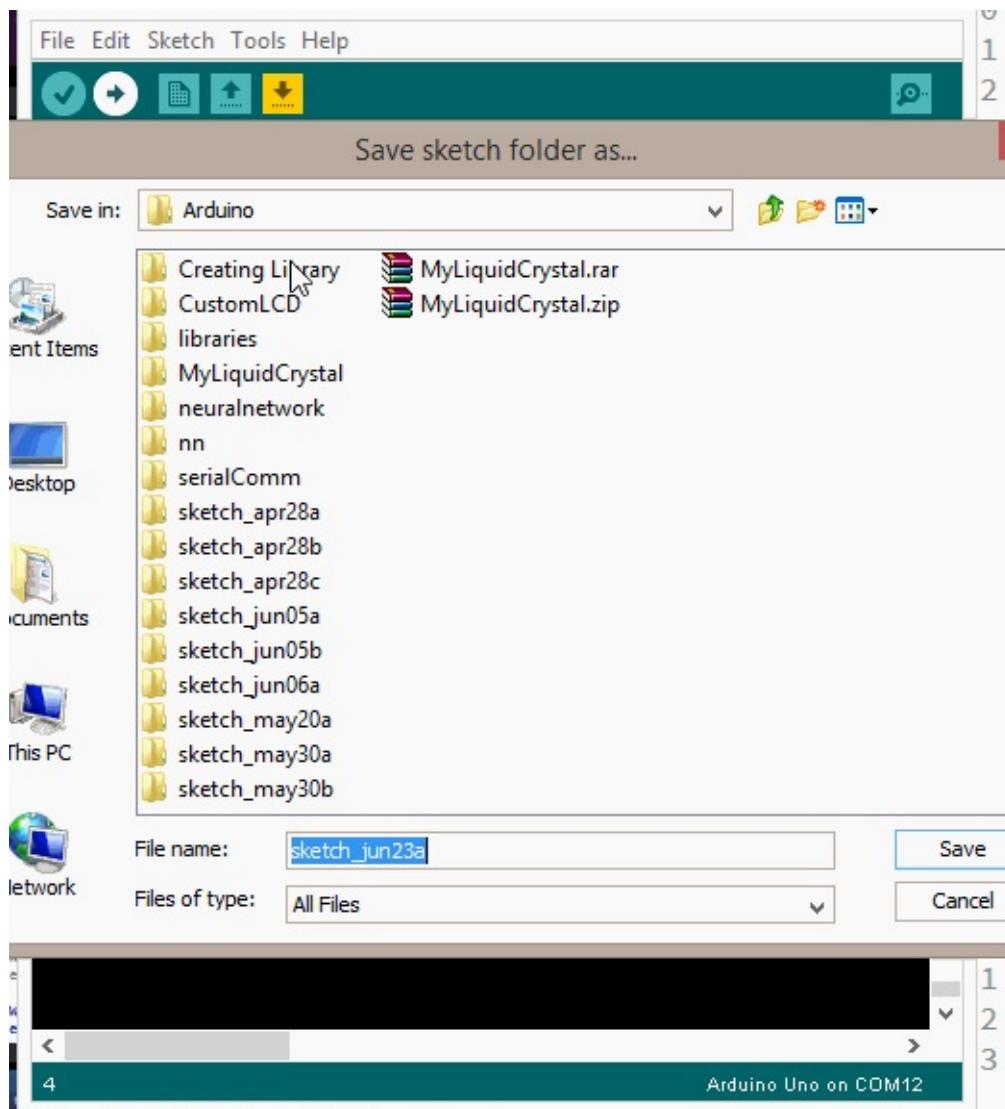
- আর্ডুইনো সাইডের কোড। সিম্পল LED জুলা নেভার কোড, [সিরিয়াল কম্যুনিকেশনে Serial](#) এর মেথডগুলো সম্পর্কে জানা যাবে। কোডটি আর্ডুইনোতে আপ্লোড করুন ও কানেক্ট করে রাখুন। তারপর Qt এ প্রোগ্রামটি রান করুন:

```
//Arduino Side
// Baud Rate 9600
int led = 13;
char command;

void setup() {
    Serial.begin(9600);
    pinMode(led, OUTPUT);
}

void loop() {
    if (Serial.available()){
        command = Serial.read();
        if (command == 'a') digitalWrite(led, HIGH);
        else digitalWrite(led, LOW);
    }
}
```

- আর্ডুইনোতে কোড আপ্লোড:



- আর্ডুইনো ঠিকভাবে কানেক্টেড হলে এইরকম একটা মেসেজ দেখতে পারবেন, আর Led অন ও অফ বাটনে ক্লিক করলে অফ অন দেখতে পারবেন

```

26 }
27
28 void MainWindow::on_ledOff_clicked()
29 {
30     qDebug() << "Led is off" << "\n";
31     ui->ledStatus->setText("Led is Off");
32     arduino->write("s");
33 }
34
35 void MainWindow::on_connectArduino_clicked()
36 {
37     arduino->close();
38     if (arduino->open(QIODevice::ReadWrite)){

```

Application Output

QArduino

C:\Users\Manash\Documents\build-QArduino-Desktop_Qt_5_4_0_MinGW_32bit-Debug\debug\QArduino.exe exited with code 0

Starting C:\Users\Manash\Documents\build-QArduino-Desktop_Qt_5_4_0_MinGW_32bit-Debug\debug\QArduino.exe...
Led is on

C:\Users\Manash\Documents\build-QArduino-Desktop_Qt_5_4_0_MinGW_32bit-Debug\debug\QArduino.exe exited with code 0

Starting C:\Users\Manash\Documents\build-QArduino-Desktop_Qt_5_4_0_MinGW_32bit-Debug\debug\QArduino.exe...
C:\Users\Manash\Documents\build-QArduino-Desktop_Qt_5_4_0_MinGW_32bit-Debug\debug\QArduino.exe exited with code 0

Starting C:\Users\Manash\Documents\build-QArduino-Desktop_Qt_5_4_0_MinGW_32bit-Debug\debug\QArduino.exe...
C:\Users\Manash\Documents\build-QArduino-Desktop_Qt_5_4_0_MinGW_32bit-Debug\debug\QArduino.exe exited with code 0

আবারও বলা হচ্ছে, আলোচনা শর্ট ও সিম্পল রাখার জন্য এই অ্যাপ তৈরি করা হয়েছে। ইউজার ইটারফেস ও প্রোগ্রাম ডিজাইনিংয়ে মনযোগ না দিয়ে প্রসিডিউরের উপরে জোড় দেওয়া হয়েছে। পরবর্তী পর্বে আমরা দেখব Arduino কে LCD এর সাথে ইটারফেস করে Qt Desktop App এর মাধ্যমে Wirelessly মেসেজ শো করানো যায়। প্রচুর F.A.Q থাকতে পারে ভেবেই সেগুলো অ্যাড করা হল না। এইখানে যে কম্প্যুনিকেশন দেখানো হল সেটা আপাতত one way। সিগনাল ও স্লট ব্যবহার করে duplex কম্প্যুনিকেশনও দেখানো হবে পরবর্তীতে।

সম্পূর্ণ প্রজেক্ট কোড পাওয়া যাবে [এখানে](#)

পাইথনে আডুইনোর সাথে সিরিয়াল কমিউনিকেশন ও আডুইনোর সিরিয়াল ইভেন্ট পরিচিতি

পাইথনে সিরিয়াল কম্যুনিকেশনের জন্য আমরা ব্যবহার করব পাইথনের `PySerial` লাইব্রেরি।

যা যা লাগবে:

- পাইথন জানতে হবে - না জানলে [এই কোস্টি করুন](#)
 - পাইথন ২.৭ - আপনার পিসিতে পাইথন না সেটাপ দেয়া থাকলে [এখান](#) থেকে দেখে নিন।
 - একটি আডুইনো বোর্ড
 - নেট কানেকশন বা `PySerial` লাইব্রেরি
 - `PyQt4` [optional]
-

পাইসিরিয়াল সেটাপ দেব আমরা `pip` পাইথন প্যাকেজ ইন্সটলারের মাধ্যমে। আপনার পিসিতে যদি `pip` ইন্সটলড না করা থাকে তাহলে [এখানে](#) দেখুন।

পাইসিরিয়াল ইন্সটল করার জন্য টার্মিনাল(লিনাক্স) বা কমান্ড প্রম্পট (উইন্ডোজ) ওপেন করে নিচের কমান্ডটি রান করুন

```
pip install pyserial
```

কিছুক্ষণের মধ্যে পাইসিরিয়াল সেটাপ হয়ে যাবে।

পাইথন দিয়ে সিরিয়াল কম্যুনিকেশন অন্যান্য ল্যাঙ্গুয়েজ দিয়ে কমিউনিকেট করার মতই। নিচে আডুইনোর কোডটা দেখা যাক,

```
#define led 13
#define baud 9600

void setup() {
    pinMode(led, OUTPUT);
    Serial.begin(baud);
}

void loop() { /*Nothing to do*/ }

void serialEvent(){
    if (Serial.available() > 0){
        //Reading string until '\n' is encountered
        String command = Serial.readStringUntil('\n');

        //Sending a reply after executing the function
        if (command.equals("on")) { digitalWrite(led, HIGH); Serial.println("LED ON"); }
        else if (command.equals("off")) { digitalWrite(led, LOW); Serial.println("LED OFF"); }
    }
}
```

লাইন ১ থেকে ৭

আগের আডুইনো পর্যালি দেখে নিন।

লাইন ৯ - **void loop() {}**

আমরা সিরিয়াল কম্যুনিকেশন হ্যান্ডেল করব `serialEvent()` নামক ইন্টারাপ্ট হ্যান্ডলার ফাংশন দিয়ে। লুপে কিছু না লিখলেও এই ইন্টারাপ্ট ফাংশনের মাধ্যমে আমরা কিভাবে কম্যুনিকেট করব সেটাই দেখানো মুখ্য উদ্দেশ্য।

লাইন ১১ - **void serialEvent() {**

সিরিয়াল ইভেন্ট ফাংশনের শুরু। এটা একটা ইন্টারাপ্ট ফাংশন। মাইক্রোকন্ট্রোলার/মাইক্রোপ্রসেসরে ইন্টারাপ্ট থাকেই। ইন্টারাপ্ট আসলে এমন ধরণের সিগনাল যেটার প্রায়োরিটি নরমাল ফাংশনের চেয়ে বেশি।

একটা উদাহরণের মাধ্যমে ব্যাখ্যা করা যাক, ধরুন আপনি বই পড়ছেন এবং পাশে আপনার মোবাইল। এখন যদি মোবাইলে কোন Call আসে তাহলে মোবাইলটি বেজে উঠবে, সাধারণত যে কেউ বই পড়া বন্ধ করে মোবাইলে কথা সেবে তারপর বই পড়বে।

এখনে মোবাইলের বিংয়িং কে আমরা `Interrupt Signal` বলতে পারি এবং আপনি যে কথা বললেন সেটাকে আমরা `Interrupt Function` বলতে পারি।

ঠিক তেমনই, আডুইনোতে সিরিয়ালে কোন ডেটা আসে তাহলে ইন্টারাপ্ট টেবিলের প্রায়োরিটি লিস্ট অনুযায়ী `serialEvent()` ফাংশনটি অটো কল হয়।

লাইন ১৮ - **String command = Serial.readStringUntil('\n')**

এর অর্থ হচ্ছে যতক্ষণ না পর্যন্ত আডুইনো \n ক্যারেক্টরটি ডিটেক্ট করছে, ততক্ষণ পর্যন্ত সে সিরিয়ালের ডেটা রিড করে `command` নামের `String` ভ্যারিয়েবলে রাখছে।

লাইন ১৭ - **if (command.equals("on")) ...**

`equals` হল `String` ক্লাসের একটা ফাংশন। যেহেতু `command` হল `String` ক্লাসের অবজেক্ট তাই আমরা `command.equals(another_string)` এভাবে কল করতে পারি।

`equals` ফাংশনের ইনপুট প্যারামিটার হল আরেকটি স্ট্রিং, যেটাৰ সাথে তুলনা কৰা হবে। `equals` এর রিটার্ন টাইপ `boolean`। অর্থাৎ, দুইটা স্ট্রিংয়ের প্রতিটি ক্যারেক্টৰ যদি সমান হয় তাহলে `equals` রিটার্ন করে `true` অথবা এটা `false` রিটার্ন করবে।

digitalWrite ... ; Serial.println ...

যদি আডুইনো `on` স্ট্রিং টা রিসিভ করে তাহলে সে `13 no LED` অন কৰবে এবং একটা রিপ্রাই দিয়ে দেবে যে `LED` অন হয়েছে।

আৰ যদি `off` রিসিভ কৰে.... এটা আশা কৰি বলা লাগবে না। :P

পাইথন প্রোগ্রাম

আমরা এতক্ষণ আডুইনো প্রোগ্রাম নিয়ে আলোচনা কৱলাম, এবাৰ একটু পাইথন প্রোগ্রাম দেখা যাক।

```

# Importing the pyserial library
import serial

# My arduino is connected to COM3, check device manager for yours
arduino = serial.Serial("COM3")

# Closing the arduino serialport just in case
arduino.close()

# Setting baudrate to 9600
arduino.baudrate = 9600

# If not open then try to open it
if not arduino.is_open:
    arduino.open()

# While the port is opened, play with it by sending string commands
while True:
    command = raw_input('Enter command: ')
    arduino.write(command + '\n')
    print arduino.readline()

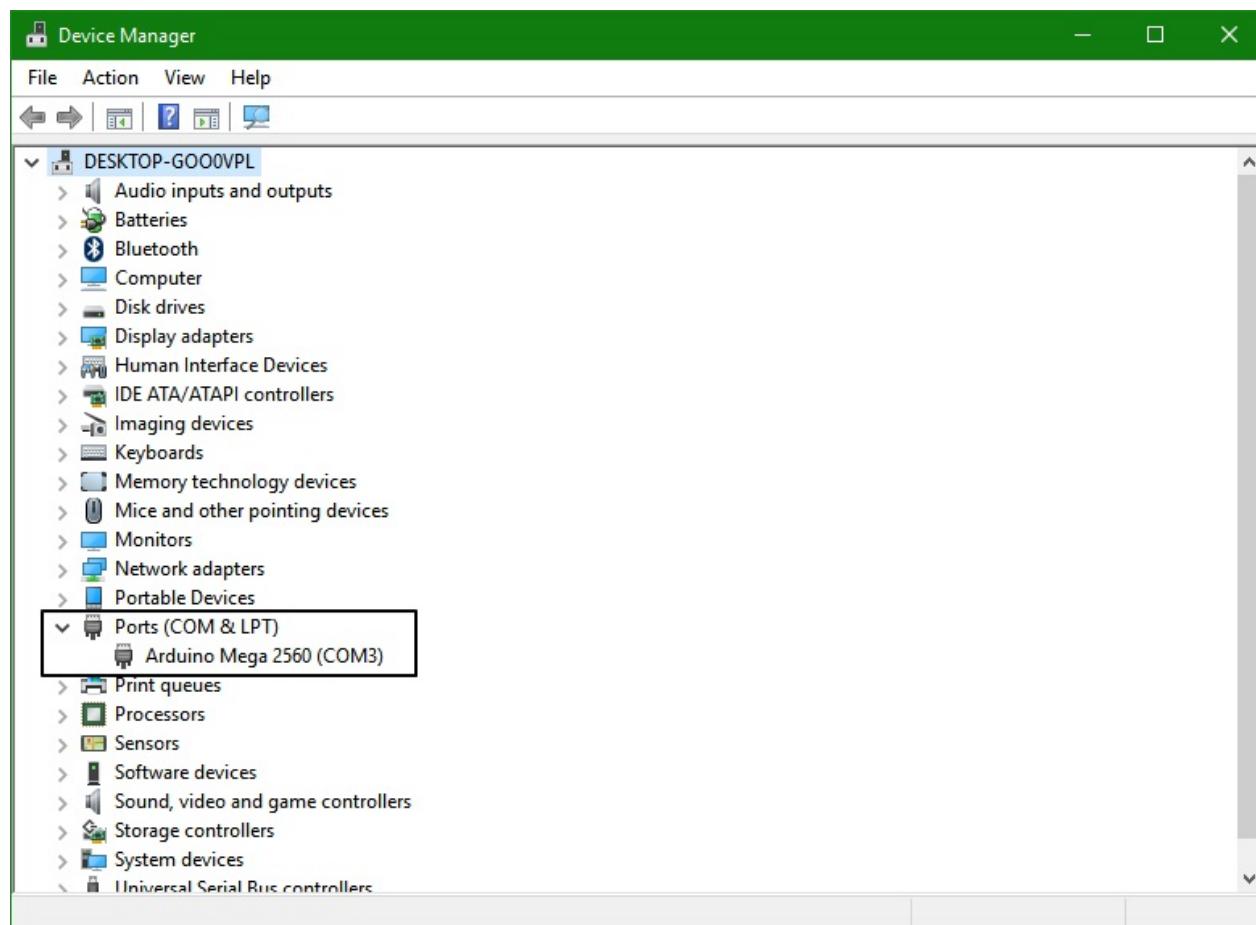
```

লাইন ২ - import serial

লাইনের ইম্পোর্ট করলাম।

লাইন ৫ - arduino = serial.Serial(com_port)

সিরিয়াল স্লাস দিয়ে `Serial` টাইপ অবজেক্ট তৈরি করলাম ও ইনপুট প্যারামিটার দিলাম আমার পিসির যে পোর্টে `Arduino` কানেক্টেড হয়েছে। অর্থাৎ আমার পিসিতে কানেক্ট হয়েছে `com3` তে, আপনার পিসি যদি উইন্ডোজ হয় তাহলে নতুন আডুইনো IDE ওপেন করেই দেখতে পাবেন, অথবা `Device manager > Ports` এ গেলেও দেখতে পাবেন (আডুইনো অবশ্যই কানেক্টেড থাকতে হবে)



লাইন ৮ - `arduino.close()`

অনেক সময় আডুইনো পোর্ট কারণে অকারণে বিজি থাকে, সেটা বিজি থাকুক কি না থাকুক, তাকে ফ্রি করার জন্য আমরা ফাংশনটা কল করলাম। এতে করে পোর্ট যদি বিজি ও থাকে তাহলেও সেটা ফ্রি হয়ে যাবে। Safety check হিসেবেও নেয়া যেতে পারে।

লাইন ১১ - `arduino.baudrate = 9600`

আডুইনো কোডে নিচ্যয়ই দেখেছেন, আমি বডরেট 9600 দিয়েছি, তাই পাইথনেও একই স্পিড রাখলাম।

লাইন ১৪-১৫ - `if not ...`

যদি আডুইনো পোর্ট ওপেন না থাকে তাহলে ওপেন করব। (ডেটা রিড রাইটের জন্য)

লাইন ১৮ - `while True:`

আমরা বেশ কিছু স্ট্রিং পাঠাতে চাচ্ছি তাই ইনফিনিট লুপ দিয়ে ভেতরের প্রোগ্রামটা লিখছি।

লাইন ১৯ - `command = raw_input('Enter command: ')`

বেসিক পাইথন জানলে এটা অবশ্যই জানবেন, না জানলে বলি, কনসোল থেকে ইনপুট নেওয়ার জন্যই এই স্টেটমেন্ট।

লাইন ২০ - `arduino.write(command + '\n')`

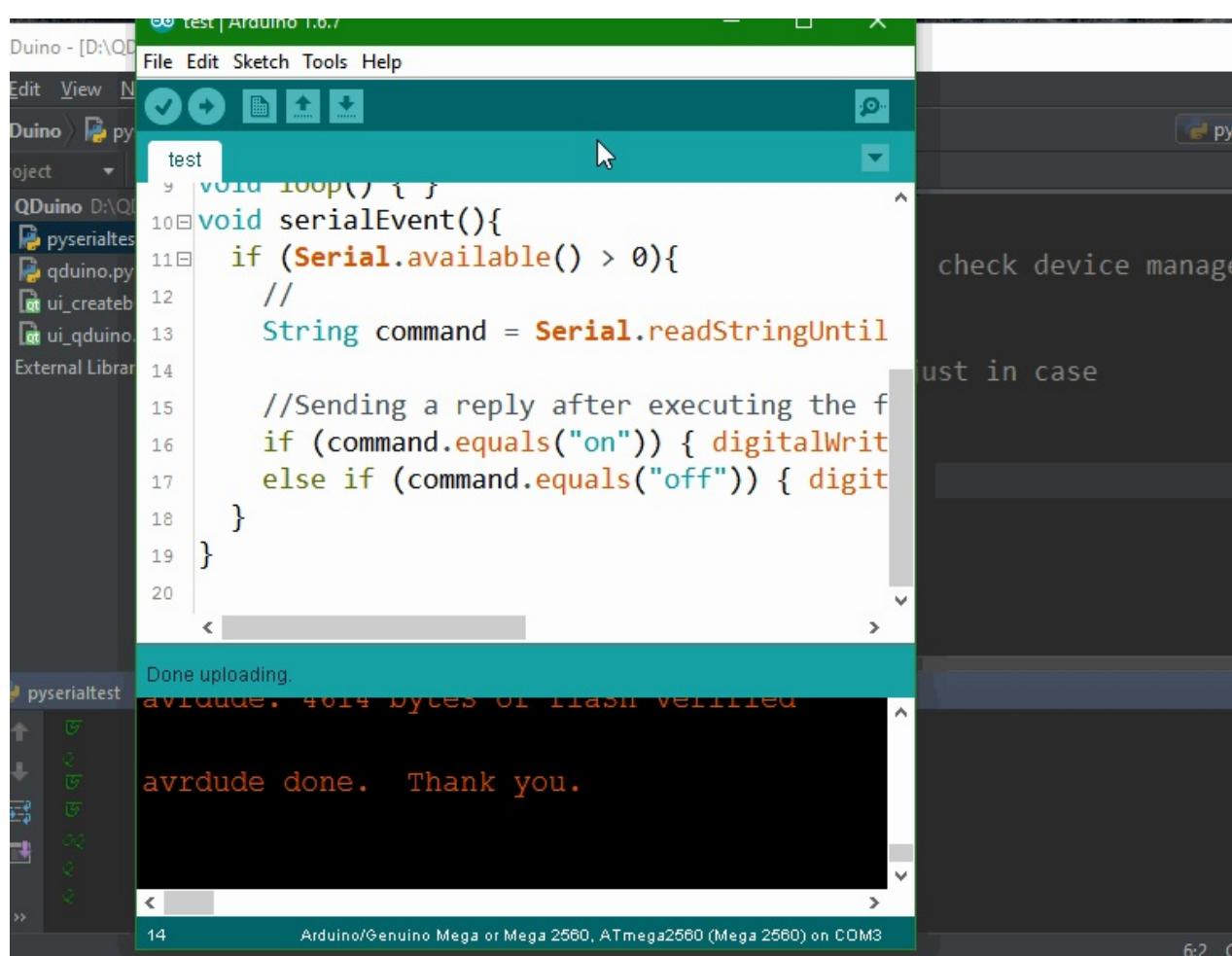
সিরিয়ালে ডেটা রাইট করার জন্য আমরা `write` ফাংশনটি কল করেছি। আর ভিতরে কনসোলে ইনপুট দেয়া স্ট্রিংটি পাঠাচ্ছি।

শেষে `+ '\n'` দেওয়ার উদ্দেশ্য হল, আডুইনো সাইডে কোড লেখার সময় আমরা শর্ত দিয়েছিলাম `readStringUntil('\n')`, তাই স্ট্রিং পাঠানোর আগে একটা 'ডেলিমিটার' জুড়ে দিচ্ছি। আপনি `\n` এর বদলে `;, -, +` যেটা খুশি সেটাই বসাতে পারবেন, শর্ত হল পাইথন থেকে স্ট্রিং পাঠানোর সময় যেন ডেলিমিটারটা স্ট্রিংয়ের শেষে থাকে এবং আডুইনোতে `readStringUntil(delimiter)` ডেলিমিটার একই থাকে।

লাইন ২১ - `print arduino.readline()`

আডুইনো যদি কোন কিছু পাঠায় তাহলে সেটা প্রিন্ট হবে।

টেস্টিং



```

void loop() {
  void serialEvent(){
    if (Serial.available() > 0){
      String command = Serial.readStringUntil('\n');
      //Sending a reply after executing the function
      if (command.equals("on")) { digitalWrite(13, HIGH); }
      else if (command.equals("off")) { digitalWrite(13, LOW); }
    }
  }
}

Done uploading.
avrduude: 4014 bytes of flash verified

avrduude done. Thank you.
  
```

PyQt4 দিয়ে আডুইনোর জন্য অ্যাপ্লিকেশন তৈরি

আপনার যদি PyQt4 সম্পর্কে ধারণা থাকে তাহলে এই কোড মডিফাই করে খুব সহজেই আপনার কাজের জন্য পাইথন দিয়ে আডুইনোর গুই তৈরি করতে পারবেন। [সম্পূর্ণ প্রজেক্ট পাবেন এখানে](#)।

এখানে আডুইনো সাইডের কোড উপরের দেওয়া কোডটাই ব্যবহার করা হয়েছে।

অ্যাপ্লিকেশন ডেমো

```

class PyDuino(PyduinoDialog, QtGui.QDialog):
    def __init__(self):
        PyduinoDialog.__init__(self)
        QtGui.QDialog.__init__(self)
        self.setupUi(self)

        # baud rate list
        self.bauds = ['1200', '2400', '4800', '9600']

        # Initializing com ports
        for device in list_ports.comports():
            self.COMComboBox.addItem(device.device)

        # adding the bauds
        for baud in self.bauds:
            self.BaudComboBox.addItem(str(baud))

```

PySerial

এখানে বেসিক কিছু আইডিয়া দেয়া হল, ভালভাবে জানতে দেখুন [অফিশিয়াল ডকুমেন্টেশন](#)।