De La Rosa Joshua 5AI 07/06/2014
Mastroeni Fabrizio

Documentazione sul progetto di fine anno: Metropolitana Automatizzata

1) Scopo prefissato

Lo scopo del progetto è verificare la possibilità di controllare in automatismo o da remoto un semplice sistema di trasporto (come per esempio una metropolitana).
Il sistema da realizzare verrà fatto funzionare in modo completamente automatizzato e/o controllato tramite un'applicazione su computer sul quale in ogni caso è possibile rappresentare lo stato del sistema di trasporto tramite un quadro sinottico. Per il progetto in questione verrà usata la scheda Arduino, una scheda di controllo open source, il cui funzionamento viene stabilito tramite un codice, che ripete tali istruzioni infinite volte in un ciclo infinito

2) Dimensionamenti delle scelte attuate

Per sperimentare le possibilità del sistema abbiamo scelto di realizzare un modello contenente almeno uno scambio e un numero limitato di 4 fermate. In questo modo è stato possibile verificare le problematiche informatiche e di controllo senza doverci sobbarcare il ripetersi di eccessivi problemi di tipo meccanico.
Partendo da un modellino di trenino alimentato a batteria, per evitare i problemi legati alla trasmissione del controllo ad un mezzo mobile, abbiamo scelto di controllare direttamente la linea di alimentazione per cui abbiamo rielaborato la vettura in modo da farla alimentare da una "rotaia" elettrificata controllata da Arduino.
Il sistema automatizzato è sprovvisto di funzioni legate alla sicurezza, poiché per averle occorreva l'utilizzo di più sensori che non potevamo ottenere per mancanza di tempo e soldi, anche perché le spese erano a carico degli studenti.
In ogni caso nel sistema è previsto un controllo remoto in modo da controllare il treno manualmente nel caso ci siano dei problemi eventuali durante l'esecuzione del sistema.

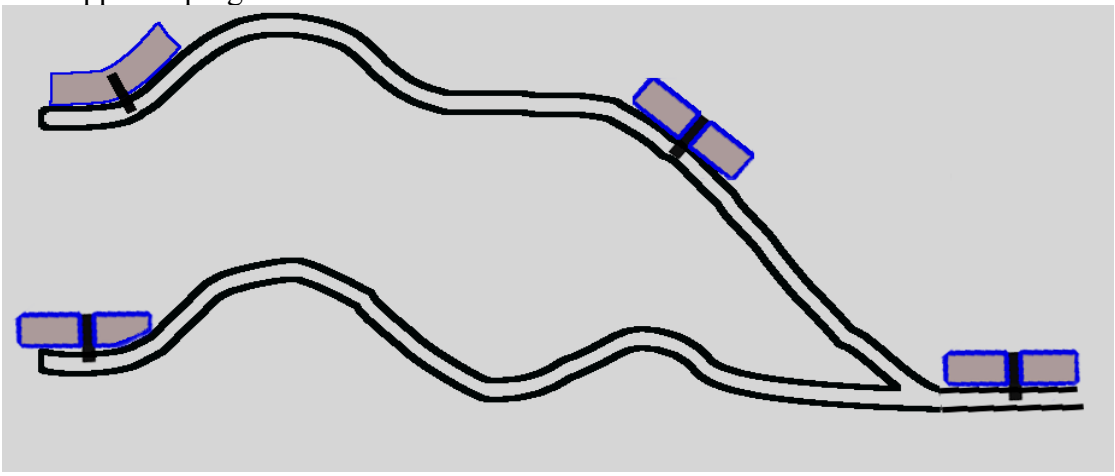
3) Funzionalità del sistema

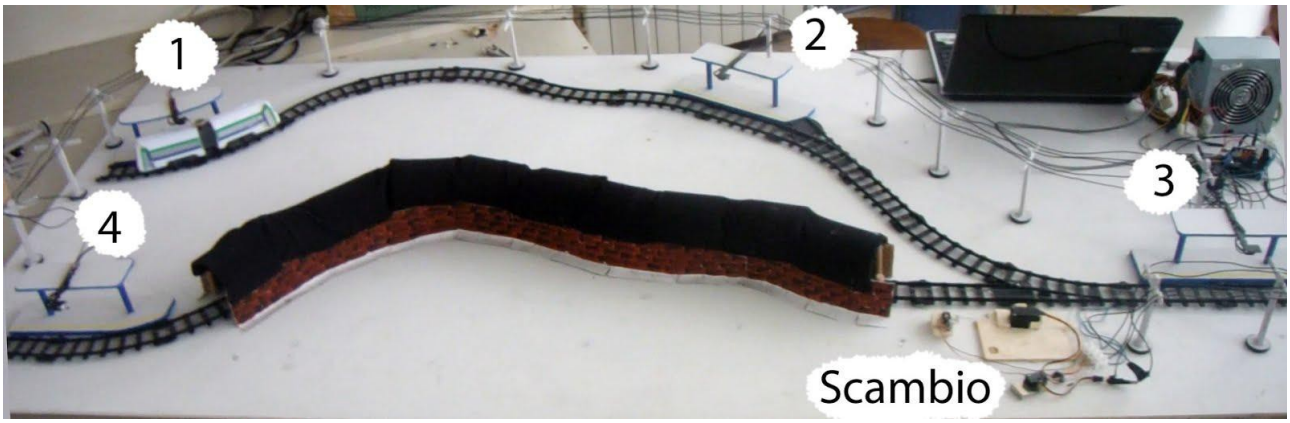
Il progetto è un modellino di sistema di trasporto automatizzato, ha un nucleo centrale che rappresenta il cuore del progetto, ciò che lo fa funzionare, che è così composto :

1. **Arduino UNO Rev 3:** Scheda input/output che contiene al suo interno il microntrollore dove è presente l'algoritmo che fa controllare il treno
2. **MotorShield Rev 3:** Un componente aggiuntivo che si colloca sopra Arduino, che ha la funzionalità di controllare un motorino passo passo o due motorini normali. Con "controllo" si intende il decidere la direzione, il voltaggio (cioè la velocità da applicare alla linea per il controllo della velocità) o l'arresto;
3. **Computer con applicazione Java:** Computer con un applicazione java dove tramite un canale di trasmissione collegato al computer e all'arduino è possibile interagire con il codice preinstallato su

Arduino Uno, in modo da poter controllare da remoto il mezzo di trasporto, vedere quale stazione occupa e quale parte di percorso sta percorrendo.

La mappa del progetto:



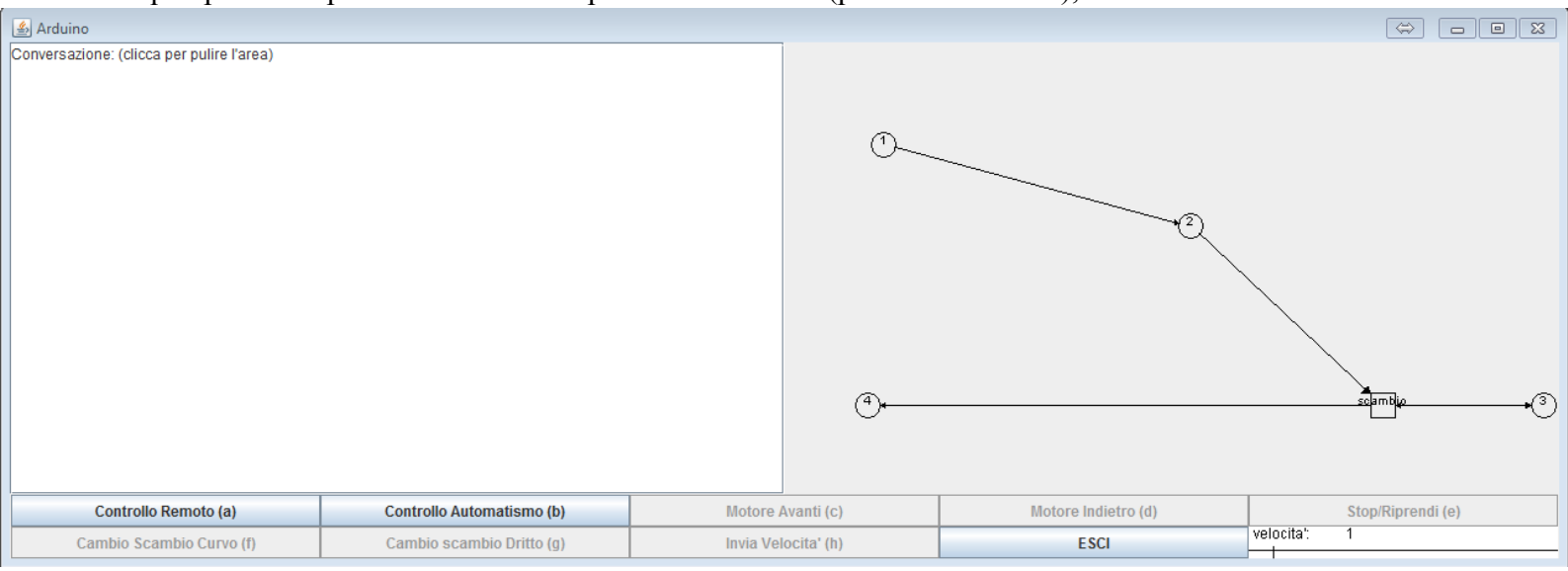


Il sistema è in grado di rilevare quando il treno arriva ad una stazione, perché ogni stazione è munita di sensore di hall (sensibile al cambiamento del campo magnetico a corto raggio) e il treno ha incorporato un magnete molto potente.
Tra le stazioni 2 e 3 e tra le stazioni 3 e 4 c'è uno scambio di cui si è riusciti a controllare il funzionamento, così la metrò può sviluppare due percorsi alternativi.
Il sistema è in grado di funzionare in maniera autonoma controllando il percorso del treno in modo che vada in entrambi i sensi di percorrenza dalla stazione 1 alla stazione 4 fermandosi per un tempo prefissato ad ogni stazione e invertendo la direzione di marcia alla stazione 3.
E' possibile anche disattivare il funzionamento automatico controllando da computer tutte le funzionalità (direzione di marcia, velocità, apertura dello scambio) manualmente.

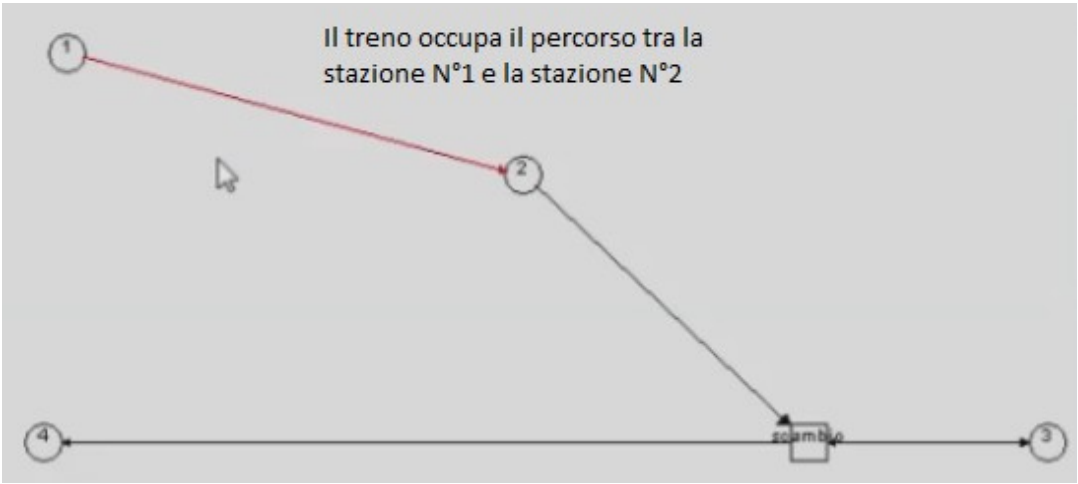
4) Manuali d'uso

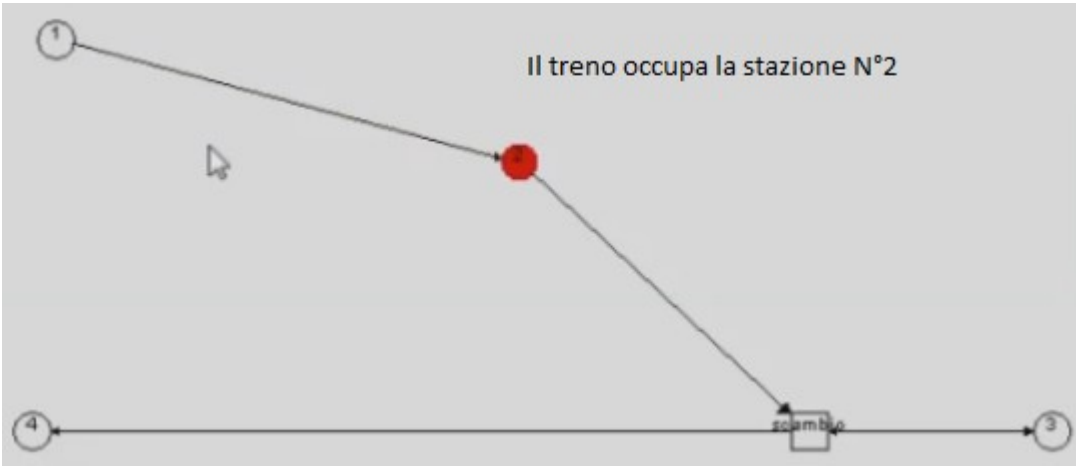
Il sistema autonomo una volta acceso fa tutto da solo, quindi ferma il treno, lo fa ripartire, gli fa cambiare direzione e muove lo scambio.

Per effettuare controlli da remoto si può collegare Arduino al computer tramite un cavo USB, e si manda in esecuzione il programma (scritto in Java) chiamato ControlArduino.
Appena inizia l'esecuzione si deve fornire la porta che identifica la connessione seriale (p.e. "COM4") su cui è registrata la connessione USB.
A quel punto si apre l'interfaccia che prevede i controlli (pulsanti e cursore), la console e il sinottico.:



Nella console, in alto a sinistra, abbiamo i messaggi che Arduino comunica al programma, nel sinottico, in alto a destra, c'è il grafo rappresentante il percorso del treno.
Quando il sinottico evidenzierà di rosso una tratta o una stazione vorrà dire che il treno occupa quella tratta/stazione.

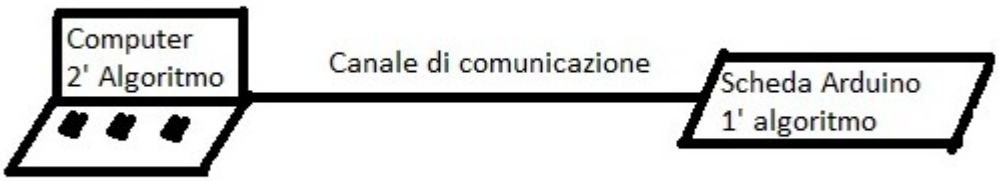




Nella sezione dei pulsanti si trovano (tra parentesi c'è il carattere che viene spedito ad Arduino):

- Controllo Remoto: interrompe l'automatismo e attiva tutti i tasti che interagiscono con il mezzo (a).
- Controllo Automatismo: interrompe il controllo remoto, disattiva tutti i tasti che modificano lo stato della metrò. (b)
- Motore avanti direzione crescente (da stazione 1 a stazione 4) (c)
- Motore indietro direzione decrescente (da stazione 4 a stazione 1) (d)
- Stop/riprendi, ferma il treno o lo fa ripartire (e)
- Scambio su/scambio giù, interagisce con lo scambio (f) (g)
- Invia velocità è una funzione che spedisce ad Arduino la velocità che il treno deve tenere, indicata dal cursore alla destra (h)

5) Strutture dati, funzionalità e Moduli presenti



Dove:

1° algoritmo scritto con il linguaggio di sviluppo di Arduino (linguaggio basto su C):

1. HallTreno_V1_7.ino: Algoritmo caricato sul microcontrollore della scheda Arduino che gestisce il funzionamento automatico del sistema e esegue i comandi ricevuti dal computer durante il controllo remoto.

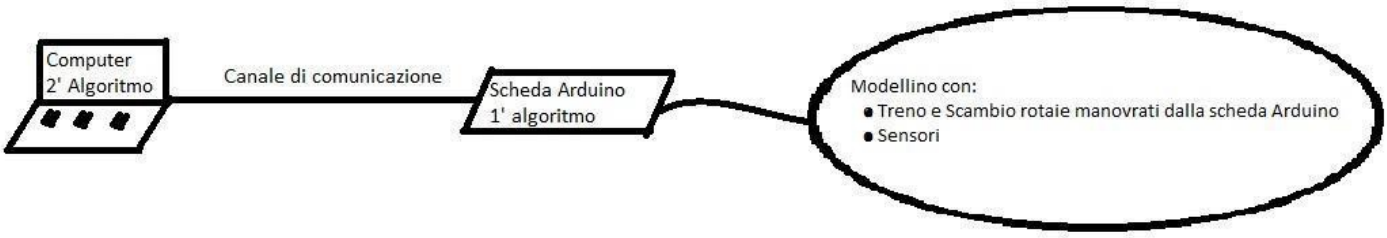
2° algoritmo scritto in Java comprendente le classi:

1. ControlArduino.java : Applicazione java che crea e gestisce l'invio e la ricezione di dati della scheda Arduino
2. Finestra.java: Algoritmo che crea la finestra grafica e gestisce gli eventi tramite i bottoni, esso richiede i seguenti file
3. Leva.java: algoritmo che crea un JPanel dove c'è una leva per regolare la velocità del treno
4. Graffetto.java: Algoritmo che gestisce un oggetto JGraphPanel dove c'è il grafo del sinottico della metropolitana, in esso sono presenti anche i metodi per colorare i percorsi e i nodi
5. Ambiente di sviluppo GraphPanel che consente di rappresentare e gestire la rappresentazione grafica di un grafo

6) Informazioni globali

Per realizzare questo programma sono state utilizzate le API di java (*JavaTM 2Platform Standard Ed. 5.0*), la documentazione dell'ambiente di sviluppo GraphPanel e la documentazione di Arduino nel sito ufficiale: http://arduino.cc/en/Reference/HomePage

7) Spiegazione dettagliata del Sistema



Nella Scheda Arduino è eseguito l'algoritmo principale dove:

1. Inizialmente vengono eseguite delle operazioni per preparare la corretta partenza del treno (funzione setup)
2. Poi viene eseguito la funzione loop all'infinito dove utilizzando una variabile chiamata autoMode stabilisce se fare eseguire la funzione dell'automatismo o no esegue o l'automatismo o il controllo remoto.
3. Poi viene eseguita la funzione loop che viene eseguita all'infinito. Utilizzando una variabile interna chiamata autoMode si stabilisce se fare eseguire il funzionamento automatico o il controllo remoto o il controllo remoto.

4. Inoltre dopo aver eseguito o l'automatismo o il controllo remoto guarda se nella linea di trasmissione è stato inviato un carattere alfanumerico dal pc, se è stato inviato e se quel carattere corrisponde al carattere 'a' allora associa alla variabile autoMode il valore false, in modo che quando verrà richiamata di nuovo la funzione loop verrà eseguita la procedura per la modalità del controllo Remoto.

Modalità automatica (funzione automatismo):

Utilizzando la variabile ContaFermate che indica il numero della prossima stazione che il treno deve raggiungere viene attivato il relativo sensore. I sensori delle stazioni sono dei sensori a effetto hall quindi trasduttori che restituiscono un valore a seconda se un campo magnetico è vicino o lontano, che restituiscono un valore che si modifica quando un campo magnetico si avvicina. Quando non c'è campo magnetico il sensore restituisce un valore intorno a 512, e quindi quando il sensore rileva un valore inferiore a 509 o superiore a 515 vuol dire che il treno è arrivato alla stazione. Se una delle due condizioni è vera viene fermato il treno, viene associato il valore false alla variabile metroMovimento, ,viene inviato al computer la posizione del treno in modo che possa aggiornare il sinottico e viene analizzata le variabili **ContaFermate** e **verso.** A seconda della situazione si farà incrementare o decrementare la variabile ContaFermate in modo da attivare il sensore della prossima stazione e attivare i controlli necessari a raggiungerla poiché ogni fermata ha qualche operazione differente da effettuare, come per esempio nella fermata 3 lo scambio deve cambiare posizione per permettere al treno di raggiungere la fermata successiva.

Più precisamente quando la variabile verso sarà true allora ContaFermate verrà incrementata, mentre quando sarà false verrà decrementata.

Operazioni da eseguire a seconda dei controlli	
Quando il treno è nella fermata N° 1	Viene associato il valore true alla variabile verso e viene inviato al computer il cambiamento della variabile tramite una stringa la quale farà capire al computer di aggiornare il sinottico Viene modificata la direzione del treno per farlo andare nella stazione N°2
Quando il treno è nella fermata N°2 e il verso coincide con il valore true	Viene modificato lo scambio per garantire il passaggio tra la stazione N°2 e la stazione N°3
Quando il treno è nella fermata N°3	Viene modificato lo scambio e la direzione del treno per garantire il passaggio del treno tra la stazione N°3 e la stazione N°4 quando il verso coincide con true, altrimenti se coincide con false viene modificato lo scambio e la direzione del treno per garantire il passaggio del treno tra la stazione N°3 e N°4

Quando il treno è nella fermata N°4	Viene associato il valore false alla variabile verso e viene inviato al computer il cambiamento della variabile tramite una stringa la quale farà capire al computer di aggiornare il sinottico Viene modificata la direzione del treno per farlo andare nella stazione N°3

Modalità Controllo remoto:

La modalità controllo remoto prevede una comunicazione tra il primo algoritmo e il secondo algoritmo, in pratica il secondo algoritmo situato nel computer invia un carattere alfa numerico al primo algoritmo che quando lo leggerà eseguirà una o più operazioni a seconda del carattere inviato.

Come fa il secondo algoritmo a inviare i comandi:

il secondo algoritmo ha una finestra grafica con dei bottoni, a seconda di che bottone viene premuto l'algoritmo invia un carattere all'arduino e l'arduino esegue un operazione a seconda del carattere inviato.

N.B: le lettere tra parentesi indicano il carattere che verrà inviato al premere di un determinato bottone

Cosa invia il secondo algoritmo e cosa fa il primo algoritmo:

Comando inviato dal secondo algoritmo (Java)	Operazione che fa il primo algoritmo (Arduino)
a	Associa a false la variabile autoMode in modo che nel loop si possa eseguire la procedura del controllo remoto
b	Associa a true la variabile autoMode in modo che nel loop si possa eseguira la procedura dell'Automatismo
c	Cambia la direzione del treno e lo fa andare verso la fermata numero uno
d	Cambia la direzione del treno e lo fa andare verso la fermata numero 4
e	Ferma il treno se il treno è in movimento o lo fa andare se è fermo
f	Fa muovere lo scambio verso su
g	Fa muovere lo scambio verso giu

h	Invia la velocità accanto al bottone esci

8) Specifiche delle Classi e dello sketch Arduino

Sketch del 1° Algoritmo, situato dentro Arduino

HallTreno_V1_7.ino **Variabili Principali:**	
int pinDirA=12; //A motore treno	Variabile che indica su cui pin operare per modificare la direzione del motore situato nel treno
int pinVelA=3; // B motore scambio	Variabile che indica su cui pin operare per modificare la velocità del motore situato nel treno
int pinBrakeA=9;	Variabile che indica su cui pin operare per interrompere o dare alimentazione al motore del treno
int pinDirB=13;	Variabile che indica su cui pin operare per modificare la direzione del motore situato nello scambio
int pinVelB=11;	Variabile che indica su cui pin operare per modificare la velocità del motore situato nello scambio
int pinBrakeB=8;	Variabile che indica su cui pin operare per interrompere o dare alimentazione al motore del treno
int vel=200;	Variabile che stabilisce la velocità del treno
int contaFermate=2;	Variabile che stabilisce la prossima stazione del treno
int hall=0;	Variabile che stabilisce quale sensore attivare
char comando;	Variabile che stabilisce l'ultimo comando di tipo char che ha inviato il computer
boolean autoMode;	Variabile che stabilisce se il controllo Automatico è attivo o no

boolean verso=true;	Variabile che stabilisce la direzione della metro
boolean metroMovimento;	Variabile che stabilisce se la metropolitana è in movimento o no, quindi se sta andando in un'altra fermata o è ferma alla fermata
int controlloFermate;	Variabile che stabilisce se chiamare la funzione ControllaFermate o no
int gradiSu=100; int gradiGiu=0;	Variabili che indicano di quanto lo scambio deve spostarsi per aprirsi e per chiudersi
Funzioni Principali:	
void setup()	Metodo che esegue le operazioni iniziali, quindi la preparazione della linea di comunicazione, l'assegnazione output dei pin per interagire coi motorini, l'invio della posizione della metropolitana e la partenza della metropolitana
void loop()	Metodo che esegue il suo contenuto all'infinito, controlla la variabile AutoMode e se è vera chiama la funzione del sistema in modalità automatizzata (automatismo) e se è falsa controlla la variabile ControlloFermata che se è falsa chiama la funzione del controllo remoto (ControlloRemoto) e se è vera chiama la funzione che Controlla in che fermata è il treno ControlloFermate e chiama la funzione del controllo Remoto
void scambio(int gradi)	Fa muovere lo scambio di tot gradi a seconda del parametro
void automatismo()	Gestisce l'automatismo della metropolitana inviando delle informazioni di tipo String al Computer per far aggiornare il Sinottico.
void ControlloRemoto()	Associa alla variabile comando l'ultima informazione di tipo Char inviata dal pc nella linea di trasmissione e a seconda di che lettera corrisponde fa un operazione diversa
int dimmiNumero()	

int potenza(int base,int esponente)	Funzione che ritorna un numero prendendo dalla linea di comunicazione prima il numero delle cifre e poi le cifre in sequenza, all'interno è chiamata la funzione potenza per comporre un numero con più cifre Esegue la potenza di un numero dando come parametro il numero e il suo esponente (chiamata nella funzione dimmiNumero per creare il numero)
void controlloFermata()	Funzione che associa alla variabile contaFermate il numero della stazione dove è presente il treno utilizzando i sensori di hall se il treno è in una fermata

1 ***Classe principale del 2° algoritmo, situato dentro il Computer:***

ControlArduino.java **Variabili principali:**	
Scanner scanner=new Scanner(System.in); SerialPort serialPort; //dichiaro un oggetto SerialPort	SerialPort, porta virtuale
BufferedReader input;	Oggetto per leggere le Stringhe che ricevo dal mio arduino
OutputStream output;	oggetto per inviare Stringhe al mio arduino
private static final int TIME_OUT	tempo di attesa
private static final int DATA_RATE	Variabile che stabilisce quanti bits per secondo possono passare nella linea di trasmissione
Finestra grafica;	L'interfaccia grafica (classe spiegata in seguito)

boolean verso
 	
Costruttore della classe:
 	
public ControlArduino()
{
String car=JOptionPane.showInputDialog(null,"scrivi la porta");
this.costruisciPort(car);
 	
PrintStream out= new PrintStream(output);
 	
Thread ricevitore=new Thread(this);
ricevitore.start();
grafica= new Finestra("Arduino",this);
}
 	
Metodi della classe pricipale:
	Variabile che indica in che verso viaggia la metropolitana
	
	Creazione dell'interfaccia grafica, classe spiegata succcessivamente

public void run()	metodo del Trhead, dove viene chiamato il metodo ricevo()
public void costruisciPort(String com)	Metodo che stabilizza e imposta la comunicazione modificando le variabili principali
public synchronized void close()	Metodo che chiude la connessione della porta Seriale
public synchronized void ricevo()	Metodo che gestisce la ricezione dati dell'arduino, questo metodo legge le informazioni presenti nel canale di trasmissione e a seconda di che Stringa invia l'arduino fa delle modifiche alla finestra grafica utilizzando i metodi di quest'ultima.
void invio(char comando)	Metodo che invia dei caratteri Char all'arduino, Solitamente chiamato nel metodo actionPerformed dela Finestra Grafica
public static void main(String arg[])	Metodo Main dove viene chiamato il costruttore principale della classe

Classi Secondarie del 2° algoritmi che gestiscono l'interfaccia grafica:

Finestra.java	
Variabili Principali:	
boolean verso	Variabile che stabilisce in che direzione viaggia la metropolitana
int numeroVelocita;	Variabile che stabilisce la velocità della metropolitana
private Leva l;	Oggetto Leva
private Container c;	Contenitore per oggetti GUI
private JPanel labContenitore;	Pannello che conterrà l'area di testo delle informazioni e il sinottico (che è rappresentato dall'oggetto Graffetto dove in pratica è presente il Grafo)
private JPanel labBott;	Pannello dove ci saranno i bottoni che serviranno per controllare da remoto il treno
JTextArea areaDiTesto;	Area di testo dove ci saranno i messaggi che Arduino invierà al computer
private JButton bott[];	Matrice dei bottoni
private JPanel leva;	Pannello dove ci sarà l'oggetto Leva
Graffetto g;	Oggetto Graffetto, oggetto che mostra il GraphPanel
ControlArduino control;	Oggetto che crea e gestisce la comunicazione tra Arduino e il Computer
Costruttori della classe:	
public Finestra()	Costruttore che crea una finestra grafica di prova senza comunicazione con Arduino
public Finestra(String nome,ControlArduino applicazione)	Costruttore che crea la finestra grafica con la comunicazione con Arduino
Metodi della Classe:	

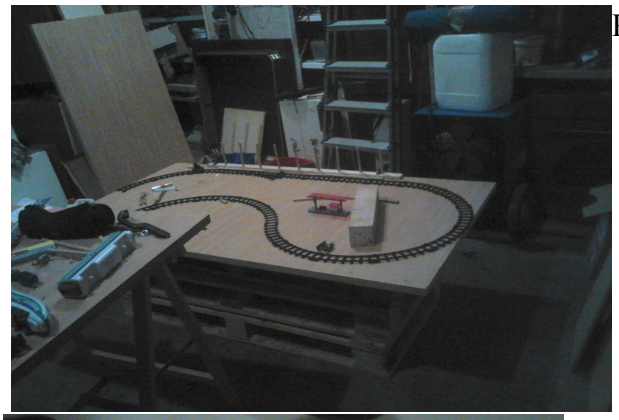
void setVerso(boolean sv)	associa alla variabile verso il parametro sv
boolean getVerso()	metodo che ritorna la variabile verso
public void actionPerformed(ActionEvent ae)	Metodo che gestisce gli eventi cioè le pressioni dei tasti nella finestra grafica, al premere di un tasto viene chiamata la funzione invio dell'oggetto control Arduino che invierà un Char, questo Char cambierà a seconda del tasto premuto
public void mouseClicked(MouseEvent e)	Metodo che gestisce gli eventi del click del mouse
public static void main(String args[])	Metodo Main che serve solo per visualizzare la finestra grafica senza comunicazione con l'arduino, quindi solo per fare un test di prova

Graffetto.java	
Variabili principali:	
GraphJPanel gr	Oggetto GraphPanel che serve a visualizzare il grafo
Costruttore della classe:	
public Graffetto()	Costruttore dove viene creato il GraphPanel e viene chiamata il metodo pulisciGrafo
void pulisciGrafo(boolean verso)	metodo con un parametro booleano che carica e adatta il grafo nel GraphPanel gr, se il parametro è vero carica il grafo che mostra il percorso che fa la metropolitana dalla fermata 1 alla fermata 4, invece se è falso carica il grafo che mostra il percorso che fa la metropolitana dalla fermata 4 alla 1
void coloraPercorso(String n,Boolean verso)	Metodo che aggiorna il grafo chiamando la funzione pulisciGrafo e colora il percorso tra 2 fermate della metro avendo come parametro la direzione della metro e la fermata attuale, (metodo chiamato dentro ControlArduino)
void coloraFermata(String nome,boolean verso)	Metodo che aggiorna il grafo chiamando la

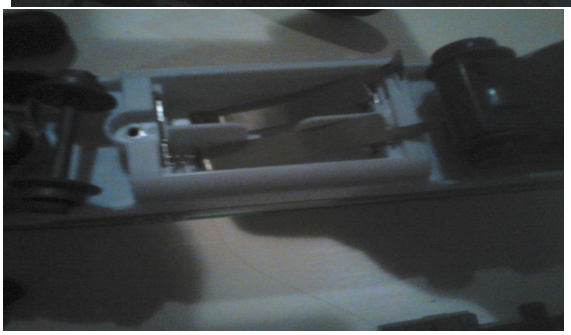
	funzione pulisciGrafo e colora la fermata avendola come parametro di tipo String

Leva.java **Variabili principali:**	
mX, My	coordinate
valore	intero che rappresenta il valore velocità(da 1 a 255)
Costruttori della classe:	
public Leva(Component, boolean)	costruttore che chiede il componente di provenienza e il valore booleano che indica se il sistema è in controllo remoto o meno e poi chiama l'altro costruttore
public void Leva()	metodo costruttore principale che chiama la superclasse esso inizia i metodi del Mouse Listener e mouseMotion Listener
Metodi principali:	
paint(Graphics g)	metodo che disegna direttamente sulla finestra/panello
mouseDragged(MouseEvent)	metodo che interviene quando il mouse si muove e clicca
mouseClicked(MouseEvent)	metodo che interviene quando muose viene cliccato(sx/dx)

9)Prove e versioni precedenti

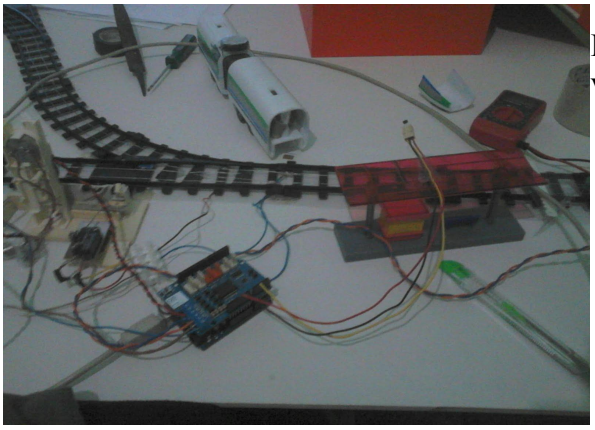


Prova treno giocattolo originale

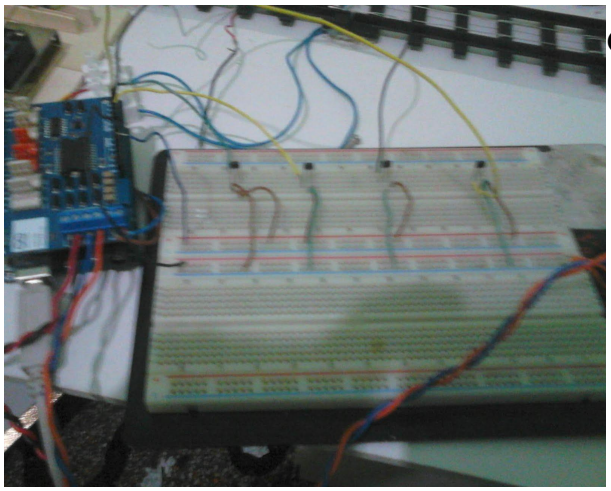


Prime idee sui contatti
parte inferiore treno

Idea vincente:
contatto molto forte e passaggio corrente netto.
Elettrificazione dei binari (filo telefonico)



Interazione con Arduino e controllo remoto prima versione.



Calibrazione dei sensori di hall (effetto mgnetico)


Stazioni in fase di montaggio


Modellino completo

10)Bibliografia esterna

Video tutorial/ dimostrazione arduino:

1. http://www.youtube.com/watch?v=b2CFKx5ASUY robot alpha nino (uso Arduino)

Altri:

1. http://scuola.arduino.cc/ lezioni di arduino in italiano e inglese
2. http://www.mauroalfieri.it/elettronica/tutorial-collega-la-motor-shield-e-pilota-i-motori.html utilizzare 2 motori con Motor shield
3. http://www.logicaprogrammabile.it/2012/pilotare-motore-dc-tramite-ponte-h/ alternativa motorshield
4. http://www.mauroalfieri.it/elettronica/frequenza-pwm-arduino-duty-cycle.html pin pwm
5. http://www.campustore.it/arduino-uno-rev3.html distributore ufficiale in Italia di arduino
6. http://www.robot-italy.com/it/arduino.html distributore ufficiale in Italia di arduino
7. http://www.electroyou.it/tardofreak/wiki/pc-e-linee-seriali spiegazione sulla possibilità di far comunicare java e un dispositivo usb (quindi pure arduino)
8. Ricerche google con parole : Modellismo fai da te , modellistica, ecc...
9. http://arduino.cc/en/Reference/HomePage Documentazione Api arduino

Indice generale