

```
////// QUELLCODE FUER PTP-2
////// Stand: 15.07.2019

////// DEFINIERUNG DER VARIABLEN
// Pinbelegung
const int LEDs [30] = {2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
14, 15, 16, 17, // Pinbelegung am Arduino
18, 19, 20, 21, 22, 23, 24, 25, 26, 27,
28, 29, 30, 31 // Pinbelegung am Arduino
};

// Weichen
int Status_Weichen [4] = {0, 0, 0, 0}; // Weichenstatus (0
-> frei; 1 -> belegt)
```

```

// Gleisfreimeldeabschnitte
int Status_Gleis1 [9] = {0, 0, 0, 0, 0, 0, 0, 0, 0}; // Gleisschrittsstatus 1 (0 -> frei; 1 -> belegt)
int Status_Gleis2 [3] = {0, 0, 0}; // Gleisschrittsstatus 2 (0 -> Frei; 1 -> Belegt)

// Rest
char State = ' '; // Character-Variabale (nimmt Wert /
Variable der seriellen Eingabe an)
int Gleis_1_Frei = 0; // Variable zur Zaehlung der freien
Gleisschritte im Gleis 1 (min. 0; max. 9)
int Gleis_1_Besetzt = 0; // Variable zur Zaehlung der
besetzen Gleisschritte im Gleis 1 (min. 0; max. 9)
int Gleis_2_Frei = 0; // Variable zur Zaehlung der freine
Gleisschritte im Gleis 2 (min 0; max. 3)
int Gleis_2_Besetzt = 0; // Variable zur Zaehlung der
besetzen Gleisschritte im Gleis 2 (min. 0; max. 3)
int W_Frei = 0; // Variable zur Zaehlung der freien Weichen

```

```
im Fahrweg (min. 0; max. 4)
int W_Besetzt = 0; // Variable zur Zaehlung der besetzen
Weichen im Fahrweg (min. 0; max. 4)

// // UNTERPROGRAMME
//void Start()
void Start() // erzeugt Unterprogramm "Start()"
{
    Serial.println("Bitte geben Sie die Fahrstraße ein!"); //
    Serielle Ausgabe des Texts
} // Ende "void Start()"

// void Grundstellung()
void Grundstellung() // erzeugt Unterprogramm
```

```

"Grundstellung()"

{
    for (int i = 0; i <= 29; i++)
    {
        digitalWrite(LEDs[i], LOW); // alle LEDs Status "LOW"
    } // Ende "for (int i = 0; i <= 29; i++)"

        digitalWrite(LEDs[1], HIGH); // LED an Pin 3 an (KS_2_Va)
        digitalWrite(LEDs[3], HIGH); // LED an Pin 5 an (Hp0_A)
        digitalWrite(LEDs[7], HIGH); // LED an Pin 9 an (Hp0_N1)
        digitalWrite(LEDs[9], HIGH); // LED an Pin 11 an (Hp0_N2)
        digitalWrite(LEDs[12], HIGH); // LED an Pin 14 an (KS_2_Va)
        digitalWrite(LEDs[14], HIGH); // LED an Pin 16 an (Hp0_F)
        digitalWrite(LEDs[18], HIGH); // LED an Pin 20 an (Hp0_P1)
        digitalWrite(LEDs[20], HIGH); // LED an Pin 22 an (Hp0_P2)

    for (int i = 0; i <= 3; i++)
    {

```

```

status_Weichen[i] = 0;           // alle Weichen freigemeldet
} // Ende "for (int i = 0; i <= 3; i++)"

for (int i = 0; i <= 8; i++)
{
    status_Gleis1[i] = 0;         // alle Gleisabschnitte im Gleis 1
freigemeldet
} // Ende "for (int i = 0; i <= 8; i++)"

for (int i = 0; i <= 2; i++)
{
    status_Gleis2[i] = 0;         // alle Gleisabschnitte im Gleis 2
freigemeldet
} // Ende "for (int i = 0; i <= 2; i++)"

} // Ende "void Grundstellung()"

```

```
// void Blinken()
void Blinken(int k) // erzeugt Unterprogramm "Blinken()";
Variable "k" muss uebergeben werden
{
    int off = 500; // Definierung Integer-Variablie "off"
(Wert: 500 "ms")
    int on = 500; // Definierung Integer-Variablie "on" (Wert:
500 "ms" )

int blinkPhase = millis() % (off + on); // Definierung
Integer-Variablie "blinkPhase" als Rest aus Division

for (int i = 1; i <= 20; i++) // fueret 20 mal aus (10x aus
& 10x an -> ca. 10s)
{
    if (blinkPhase < off) // wenn Variable kleiner als 500
"ms"
{
```

```
digitalWrite(k, LOW); // setze Zustand der LED auf "LOW"
} // Ende "if (blinkPhase < off)"

else // Variable "blinkPhase" groesser als 500 "ms"
{
    digitalWrite(k, HIGH); // setze Zustand der LED auf
    "HIGH"
} // Ende "else"

} // Ende "for (int i = 1; i <= 20; i++)"

} // Ende "void Blinken ()"

// void LEDs_on()
void LEDs_on(int k, int j) // erzeugt Unterprogramm
"LEDs_on ()"; Integer-Variablen "k" und "j" muessen uebergeben
werden
```

```

{
    for (int i = 0; i <= k - 1; i++)
    {
        digitalWrite(LEDs[j], HIGH); // Zustand der LED an Array
        Stelle "j" aus Array "LEDs" Zustand "HIGH" (Stellen folgen aus
        Werten des Arrays "benoetigte_LEDs")
    } // Ende "for (int i = 0; i <= k-1; i++)"

} // Ende "void LEDs_on()"

// Weichen_stellen()
void Weichen_stellen(int k, int j, int h) // erzeugt
Unterprogramm "Weichen_stellen()"; Integer-Variablen "k", "j"
und "h" muessen uebergeben werden
{
    for (int i = 0; i <= k - 1; i++)
    {

```

```

if (LEDs[j] == LOW) // wenn LED an Stelle "j" des Arrays
    LEDs[] Zustand "LOW" hat (Stelle folgen aus Array "Weichen_r[]" und
    "Weichen_f[]"
{
    digitalWrite(LEDs[j], HIGH); // setze Zustand LEDs[j]
    zu "HIGH"
    digitalWrite(LEDs[h], LOW); // setze Zustand LEDs[h]
    zu "HIGH"
} // Ende "if (LEDs[j] == LOW)"

} // Ende "for (int i = 0; i <= k-1; i++)"

} // Ende "void Weichen_stellen(int i = 0; i <= k; i++)"

```

```

// void Gleisfreimeldung_1()
void Gleisfreimeldung_1(int j, int k) // erzeugt
Unterprogramm "Gleisfreimeldung_1()"; Integer-Variablen "j" und

```

```

"K" muessen uebergeben werden
{
    for (int i = j; i <= k; i++)
    {
        if (Status_Gleis1[i] == 0)      // wenn Wert an Stelle "i"
des Arrays "Status_Gleis1" gleich 0
        {
            Gleis_1_Frei = Gleis_1_Frei + 1; // erhöhe Zaehler
"Gleis_1_Frei" um 1
        } // Ende "if (Status_Gleis1[i] == 0)"

    else
    {
        Gleis_1_Besetzt = Gleis_1_Besetzt + 1; // erhöhe
Zaehler "Gleis_1_Besetzt" um 1
    } // Ende "else"

} // Ende "for (int i = j; i <= k; i++)"

```

```
// Ende "void Gleisfreimeldung_1(int j, int k)"  
  
// void Gleisfreimeldung_2()  
void Gleisfreimeldung_2(int j, int k) // erzeugt  
Unterprogramm "Gleisfreimeldung_2()" ; Integer-Variablen "j" und  
"k" muessen uebergeben werden  
{  
    for (int i = j; i <= k; i++)  
    {  
        if (Status_Gleis2[i] == 0) // wenn Wert an Stelle "i"  
des Arrays "Status_Gleis2" gleich 0  
        {  
            Gleis_2_Frei = Gleis_2_Frei + 1; // erhoehe Zaehler  
"Gleis_2_Frei" um 1  
        } // Ende "if (Status_Gleis2[i] == 0)"
```

```

else
{
    Gleis_2_Besetzt = Gleis_2_Besetzt + 1; // erhoehe Zaehler
    "Gleis_2_Besetzt" um 1
} // Ende "else"

} // Ende "for (int i = j; i <= k; i++)

} // Ende "void Gleisfreimeldung_2(int j, int k)"

// void Weichenfreimeldung ()
void Weichenfreimeldung (int j, int k) // erzeugt
Unterprogramm "Weichenfreimeldung ()"; Integer-Variablen "j" und
"k" muessen uebergeben werden
{
    for (int i = j; i <= k; i++)
}

```

```

if (Status_Weichen[i] == 0) // wenn Wert an Stelle "i"
des Arrays "Status_Weichen" gleich 0
{
    W_Frei = W_Frei + 1; // erhöhe Zaehler "W_Frei" um 1
} // Ende "if (Status_Weichen[i] == 0)"

else
{
    W_Besetzt = W_Besetzt + 1; // erhöhe Zaehler
    "W_Besetzt" um 1
} // Ende "else"

} // Ende "for (int i = j; i <= k; i++)"

} // Ende "void Weichenfreimeldung()"

```

```
//// VOID SETUP()
void setup()
{
    // Serielle Verbindung
    Serial.begin(9600);

    // Unterprogramm oeffnen
    Start();

    // Definierung der LEDs als Ausgang
    for (int i = 0; i <= 29; i++)
    {
        pinMode(LEDs[i], OUTPUT);
    } // Ende "for (int i = 0; i <= 35; i++)"

Grundstellung();
```

```
 } // Ende "void setup()"

//// VOID LOOP()
void loop()
{
    if (Serial.available() > 0) // wenn serielle Eingabe
getaetigt
    {
        State = Serial.read(); // Variable "State" nimmt "Wert"
der serielle Eingabe an
        switch (State) // switch-case
        {
            // FAHSTRABE_13A.(13N1)_MIT_D1
            case 'a':

```

```

// GLEISFREIMELDEPRUEFUNG
Gleisfreimeldung_1(0, 7); // Werte an Unterprogramm
"Gleismeldung()" uebergeben

// WEICHENFREIMELDEPRUEFUNG
Weichenfreimeldung(0, 3); // Werte an Unterprogramm
"Weichenfreimeldung()" uebergeben

// EINSTELLUNG_DER_FAHRSTRAESE
if ((Gleis_1_Frei == 8) && (W_Frei == 4)) // wenn die
ben. Gleisabschnitte frei und Weichen nicht verschlossen
{
    // WEICHENLAGE_UEBERPRUEFEN_EINSTELLEN_VERSCHLIESSEN
    int Weichen_r[4] = {22, 24, 27, 29}; // LEDs fuer
richtige Weichenlage
    int Weichen_f[4] = {23, 25, 26, 28}; // LEDs fuer
falsche Weichenlage

```

```

for (int i = 0; i <= 3; i++)
{
    Weichen_stellen(4, Weichen_r[i], Weichen_f[i]);
    // Werte an Unterprogramm "Weichen_stellen()" uebergeben
} // Ende "for (int i = 0; i <= 3; i++)"

for (int i = 0; i <= 3; i++)
{
    Status_Weichen[i] = 1; // Status der
    Arraystellen 0 - 3 des Arrays "Status_Weichen[]" auf 1 erhöhen
    (Weichen verschlossen)
} // Ende "for (int i = 0; i <= 3; i++)"

// GLEISABSCHNITTE_BESETZEN
for (int i = 0; i <= 7; i++)
{
    Status_Gleis1[i] = 1; // Status der Arraystellen
    0 - 7 des Arrays "Status_Gleis1[]" auf 1 erhöhen
}

```

```
( Gleisabschnitte belegt )
} // Ende "for (int i = 0; i <= 7; i++)"

// SIGNALISIERUNG_EINSTELLEN
int benoetigte_LEDs [ 3 ] = { 0, 5, 7 } ; // LEDs fur
Signalisierung der Fahstraße
for (int i = 0; i <= 2; i++)
{
    LEDs_on (3, benoetigte_LEDs [ i ]) ; // Werte des
    Arrays "benoetigte_LEDs [ ] " an Unterprogramm "LEDs_on () "
    uebergeben
} // Ende "for (int i = 0; i <= 2; i++)"

// FAHRSTRÄE_EINGESTELLT
Serial.println ("13A. (13N1) mit D1") ; // serielle
Ausgabe des Texts

// WARTEN_BIS_FAHRSTRAßENAUFLOESUNG
```

```
delay(10000); // 10s warten

// GRUNDSTELLUNG_DER_SIGNALE
digitalWrite(LEDs[0], LOW); digitalWrite(LEDs[1],
HIGH);
// Pin 2 -> LOW; Pin 3 -> HIGH
digitalWrite(LEDs[5], LOW); digitalWrite(LEDs[3],
HIGH);
// Pin 7 -> LOW; Pin 5 -> HIGH

// GLEISABSCHNITTE_FREI_MELDEN
for (int i = 0; i <= 7; i++)
{
    Status_Gleis1[i] = 0; // Status der Arraystellen
    0 - 7 des Arrays "Status_Gleis1[]" auf 0 (Gleisabschnitte
freigemeldet)
} // Ende "for (int i = 0; i <= 7; i++)"

// ZIELGLEISABSCHNITT_BESETZEN
Status_Gleis1[4] = 1; // Status der Arraystelle 4
```

```

des Arrays "Status_Gleis1[]" auf 1 (Zielgleisabschnitt belegt)

    //***** WEICHENVERSCHLUSS_AUFHEBEN
    Status_Weichen[0] = 0;           // Status der Arraystelle 0
des Arrays "Weichen[]" auf Null (W01 freigemeldet)
    Status_Weichen[1] = 0;           // Status der Arraystelle 1
des Arrays "Weichen[]" auf Null (W03 freigemeldet)
    Status_Weichen[2] = 0;           // Status der Arraystelle 2
des Arrays "Weichen[]" auf Null (W11 freigemeldet)
    Status_Weichen[3] = 0;           // Status der Arraystelle 3
des Arrays "Weichen[]" auf Null (W13 freigemeldet)

    // MELDUNG_FAHRSTRÄFE_AUFGELÖST
    Serial.println("Fahrstraße aufgelöst!");   //
serielle Ausgabe des Texts

} // Ende if ((Gleis_1_Frei == 8) & (W_Frei == 4)) "

```

```
else
{
    Serial.println("Gleis besetzt");
    // serielle
    Ausgabe des Texts
} // Ende "else"

Gleis_1_Frei = 0; Gleis_1_Besetzt = 0; // Zaehler
auf Null setzen
Gleis_2_Frei = 0; Gleis_2_Besetzt = 0; // Zaehler
auf Null setzen
W_Frei = 0; W_Besetzt = 0; // Zaehler auf Null setzer
State = ' ';
// "setzt" Variable "State" "zurueck"
break;
// beendet switch-case -> wartet auf neue
serielle Eingabe

default: // wenn keine Uebereinstimmung mit "State"
```

```
Serial.println("Fehler"); // serielle Ausgabe des  
Texts  
    Gleis_1_Frei = 0; Gleis_1_Besetzt = 0; // Zaehler  
auf Null setzen  
    Gleis_2_Frei = 0; Gleis_2_Besetzt = 0; // Zaehler  
auf Null setzen  
    W_Frei = 0; W_Besetzt = 0; // Zaehler auf Null setzer  
State = ' '; // "setzt" Variable "State" "zurueck"  
break; // beendet switch-case -> wartet auf neue  
serielle Eingabe  
}  
// Ende "switch (State)"  
}  
// Ende "if (Serial.available() > 0)"  
}  
// Ende "void loop()"
```